LOW-POWER OPTICAL COMPUTING FOR ARTIFICIAL INTELLIGENCE AND QUANTUM APPLICATIONS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Shi-Yuan Ma

August 2024

LOW-POWER OPTICAL COMPUTING FOR ARTIFICIAL INTELLIGENCE AND QUANTUM APPLICATIONS

Shi-Yuan Ma, Ph.D.

Cornell University 2024

Computing with analog physical systems, including optical ones, has experienced a reawakening in recent years, driven by the rapid expansion of artificial intelligence. As the size of deep neural networks continues to grow, there is an increasing demand for more efficient computation methods. Neural-network computations inherently tolerate lower precision in individual steps, particularly when optimized for certain tasks, making low-precision analog computing a viable alternative to universal digital electronic computers. Unlike well-calibrated digital electronic devices, analog physical computing may offer more uncertainty, especially when operated at low power, but leverages inherent physical processes to achieve more efficient computation.

In this thesis, I summarize some of my Ph.D. work that explored energy-efficient computation through optical systems. Initially, we constructed a large-scale optical setup capable of performing linear operations with large vector sizes. This setup features substantial spatial parallelism, leveraging the optical energy advantage in computation and making many optical computing tasks experimentally feasible. Using this setup, we demonstrated an ultra-low-optical-energy implementation of optical neural networks, previously only predicted in theory. We also examined the energy scaling advantages for state-of-the-art large deep learning models, ensuring an asymptotic overall energy consumption advantage for very large models.

We then ventured beyond the scope of conventional optical computing. We investigated technologies to generate, control, and detect highly multimode quantum states, preparing to exploit their extensive quantum features for computational and sensing innovations. We also explored integrating inherent stochastic physical processes into neural network modeling, achieving significant performance gains with minimal physical resources. Additionally, we studied how this method can benefit image sensing problems under very restricted detection energy. These explorations further advance the potential for efficient computation utilizing low-power stochastic physical systems.

BIOGRAPHICAL SKETCH

In 1996, Shi-Yuan Ma was born in Lu'An, Anhui, China, his hometown. In 2014, he decided to major in physics. In 2018, he received his B.S. degree in Physics from University of Science and Technology of China, with a minor in Computer Science. Also in 2018, he joined the Applied Physics Ph.D. program at Cornell University. In 2019, he joined Professor Peter McMahon's group just as it was taking off at Cornell.

To all Cornell students, Finger Lakes, my family and friends.

ACKNOWLEDGEMENTS

First of all, a huge thanks to my advisor, Professor Peter McMahon, and the co-founders of the McMahon Lab: Professor Tatsuhiro Onodera, Professor Logan Wright, and Professor Tianyu Wang, who have all gone on to become professors themselves now. I feel that the McMahon Lab mindset, such as the research tastes and way of thinking, has seeped into me over the ~1800 days, and will be a permanent part of who I am for the rest of my life.

Peter has always been there for me, offering help and never turning things down. Honestly, I feel so lucky to have him as my advisor. Thanks to him, I fell in love in these amazing research directions, which are largely reflected in this thesis. Without his unwavering support, I wouldn't have rediscovered my passion for research and kept it alive till now, and hopefully forever. Hiro is always available to provide suggestions. He's super nice and warm-hearted, always sharing useful views and life experiences. When I first met him, I saw him as a fellow student, only to gradually realize how experienced and senior he is. He's really a cool guy. Logan is truly someone I look up to. He really is the image of the ideal researcher in my mind. Or more accurately, he made me realize that image. Throughout the years, there were many times I felt, "Wow, Logan did that. I would probably do the same if I were in that position," and he's just there, being himself. Just like a distant, joyful beacon, even if I only look up and see it occasionally, that glance always makes the current path more enjoyable, no matter if I end up reaching that light or not. For Tianyu, it's really hard to find the proper words to describe his all-encompassing influence on me. Maybe it's just like the air. I'll always proudly be his apprentice in my career. If I'm lucky enough to achieve anything good in my research journey, I'll largely owe that to him, who nurtured me in my early years when I knew

basically nothing, like water and soil for a little seed. His relentless drive for greatness, his extreme perfectionism about some things (like what makes a good researcher) and wise pragmatism about some other things (like what needs to be done today), make his character in my mind incredibly rich, and admirable.

Besides, I really appreciate the chats and discussions with my fellow labmates, which I believe are the heart and soul of research. Genuine thanks to my dear colleagues and friends Jeremie Laydevant, Alen Senanian, Sridhar Prabhu, Vladimir Kremenetski, Mandar Sohoni, Ben Malia, Valeria Cimini, Ryo Yanagimoto, Maxwell Anderson, Fan Wu, Saeed Khan, Federico Presutti, Martin Stein, and Ruomin Zhu for all the enjoyable and insightful discussions. As I wrote each name, countless memories surfaced. I'll keep those with me now rather than try to squeeze them into words here. I also want to give special thanks to Professor Zheshen Zhang, whose research I had admired for years before finally meeting him at my first major conference. His generous treatment—engaging discussions, taking me for a tour, sharing invaluable life advice—left me incredibly flattered and greatly reignited my passion for research in the later years of my Ph.D. journey.

In the end, it might be worth noting that the one who completed this dissertation is also the one who is about to conclude a six-year idyllic country life in the village of Ithaca. In this context, very briefly, I thank everything, including but not limited to the snow, stars, water, trees, and bumpy roads in upstate New York, and everyone, which and who have accompanied me for some period during this beautiful once-in-a-lifetime journey, which I will look back on and feel melancholy about at some point in the future.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

Much of the content are adapted from the work presented in Refs. [1, 2, 3, 4].

1.1 Optical Computing for Artificial Intelligence

The development and widespread use of very large neural networks for artificial intelligence [5, 6, 7] has motivated the exploration of alternative computing paradigms—including analog processing—in the hope of improving both energy efficiency and speed [8, 9]. The widespread commercial implementation of increasingly complex deep neural networks (DNNs) has resulted in rapid growth in the total energy consumption of machine learning, and in large-scale deployments, 80-90% of the cost is for inference processing [10]. The rate at which the energy requirements for state-of-the-art DNNs is growing is unsustainable and urgently needs to be addressed [7]. Both software and hardware advances are important for reducing energy consumption: DNN models (the software) can be made more efficient [7], and hardware for executing DNN models can be made more efficient, by specializing the hardware to the type of computations involved in DNN processing [11].

Optical implementations of neural networks using analog optical systems have experienced a resurgence of interest over the past several years [12, 13, 14, 15, 16, 17, 18, 1, 19, 20, 21, 22]. They have been proposed as deep-learning accelerators that can in principle achieve better energy efficiency and lower latency than electronic processors [23, 15, 24, 25, 26]. For deep learning, optical

processors' main proposed role is to implement matrix-vector multiplications [12, 13], which are typically the most computationally-intensive operations in DNNs [11].

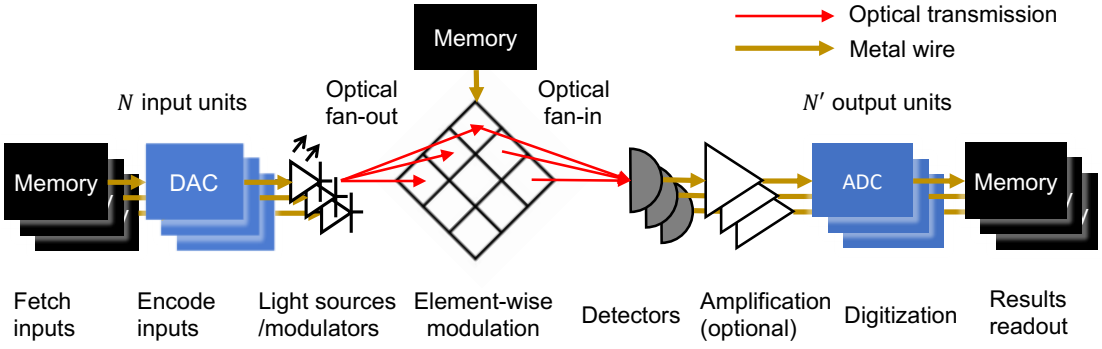


Figure 1.1: **System block diagram of an integrated optical matrix-vector multiplier.**

Theory and simulations have suggested that optical neural networks (ONNs) built using optical matrix-vector multipliers (Figure 1.1) can exhibit extreme energy efficiency surpassing even that of irreversible digital computers operating at the fundamental thermodynamic limit [24]. In order to achieve an energy advantage, an optical matrix-vector multiplier needs *massively parallel execution* of the scalar multiplications and additions that constitute a matrix-vector multiplication [25] (we will build one in Chapter 3). It has been predicted that for *sufficiently large vector sizes*, matrix-vector multiplication can be performed with an optical energy cost of less than 1 photon per scalar multiplication, assuming the standard quantum limit for noise [24] (we will demonstrate in Chapter 4).

1.2 Energy Scaling of Optical Neural Networks

Deep learning models' exponentially increasing scale is both a key driver in advancing the state-of-the-art and a cause of growing concern about their energy usage, speed, and practicality. This has led to the development of hardware accelerators and model training/compression/design techniques for efficient and fast inference on them. Because they still perform all the underlying operations using the same physical mechanisms, most digital-electronic accelerators [27, 28, 29, 30, 31] can improve performance by constant factors. While these may be large factors, they do not change the way costs scale with model compute requirements.

Analog accelerators can be different from digital ones in that the energy cost of performing computations may fundamentally scale differently than digital systems. For example, in optics or analog-electronic crossbar arrays, a common heuristic is that the energy of a matrix-vector product scales *linearly* with vector size, rather than the $\sim d^2$ of digital systems (assuming all dimensions are $\sim d$). This is a key intuition for why alternative analog computing platforms using optics have been proposed as a new paradigm for better scalability [32, 26, 15, 25, 33, 34, 23]. Ideally, the scaling is asymptotically better than digital systems in energy per MAC [24, 1, 21, 25]. This is because in existing digital systems there must be some amount of energy paid per element-wise multiplication, which does not change with the number of multiplications, so the power must scale proportionally to the number of MACs (if other overheads are ignored) [24]. By contrast, the multiplication in optics may be free; the energy cost is in encoding the data with enough photons such that the output signal-to-noise (SNR) is high enough for the final answer, regardless of operand size.

However, these optical neural networks (ONNs) have additional complexities and limitations of their own such as low precision, noise, and analog/digital data conversion overheads which depend on the access patterns of the model running. Thus, advantageously accelerating any neural network architecture with ONNs is in practice hard, and DNNs without the necessary activation statistics and model architecture may not achieve this scaling.

1.3 The Problem of Low Precision in Analog Computing

Compared to their digital counterparts, analog processors—including those constructed using optics—inevitably suffer from noise and often face issues with imperfect calibration and drift. These imperfections can degrade the accuracy of neural network inference performed using such systems [12, 35, 36, 37]. To mitigate the impact of noise, various *noise-aware training* schemes have been developed [38, 39, 40, 41, 42, 43, 44, 45]. These schemes treat the noise as a relatively small perturbation to an otherwise deterministic computation, either by explicitly modeling the noise as the addition of random variables to the processor's output or by modeling the processor as having finite bit precision. Noise-aware training is essential for the successful implementation of analog neural networks, as the models need to be resilient to the noise and errors inherent in practical analog devices (e.g., the projects discussed in Chapters 4 and 5).

The work discussed in Chapter 4 [1], along with another study published shortly after [21], demonstrates state-of-the-art ultra-low optical energy usage in optical neural networks (ONNs). These efforts have achieved notable results by using merely hundreds to thousands of photons (SNR $\lesssim 10^2$) to represent

the neuron pre-activation signal prior to photodetection (or equivalently, *<1 photon per MAC*), facilitated by neural network models specifically trained to enhance noise robustness. However, they were still in this regime where *noise is a small perturbation*. More typically, millions of photons per activation are used to achieve a sufficient SNR for reliable outcomes [17, 18, 21, 46].

Such demonstrations encounter significant difficulties if each photodetector receives only a few photons, as the uncertainty in highly stochastic activations becomes comparable to the signal itself (SNR $\sim$ 1), making noise mitigation challenging. In this extreme scenario, these highly stochastic neuron activations can no longer be regarded as essentially deterministic signals with superimposed noise. This situation appears to violate fundamental principles of computation. However, by leveraging recent innovations in machine learning modeling techniques, this seemingly insurmountable challenge can be addressed, as we will discuss in Chapter 7.

1.4 Outline of the Thesis

The outline of the thesis is as follows:

In Chapter 2, I provide some background information about neural-network computation, optical neural networks and Gaussian quantum optics.

In Chapter 3, we introduce a large-scale optical matrix-vector multiplier setup. This setup can perform dot product operations on vectors of up to approximately half a million elements, thereby leveraging the optical energy advantage through substantial parallel execution [25].

In Chapter 4, we demonstrate an optical neural network (ONN) using less than one photon per multiply-accumulate (MAC) operation while achieving over 90% test accuracy on MNIST classification. This work marks the first experimental implementation of this kind, previously only proposed in theory [24]. The success is attributed to a neural network model trained to be noise-resilient.

While Chapter 4 primarily focuses on optical energy, Chapter 5 expands the discussion to the overall energy consumption of the entire system, including both optical and electrical components. We explore the optical implementation of state-of-the-art large-scale artificial intelligence models and project an asymptotic overall energy advantage, attributed to the distinct energy scaling properties of optical systems compared to digital electrical devices.

In Chapter 6, we introduce a highly multimode spectrometer enabled by parallel single-photon detection, useful for efficiently detecting weak classical or quantum optical signals. This capability is achieved through careful calibration of photon-counting cameras and system optimization tailored to specific quantum optical applications.

In Chapter 7, we present an innovative approach to ONN modeling that differs from all previous implementations, called *physics-aware probablistic modeling*. By integrating actual stochastic physical processes and implementing the models in a probabilistic manner, this approach significantly reduces the requirements for optical energy consumption (or equivalently, signal-to-noise ratios) in ONN inference. It also has potential compatibility with quantum physical systems, such as those introduced in Chapter 6.

Building on Chapter 7, Chapter 8 explores a more practical and near-term

application of this approach. We demonstrate that by optimizing an operation in the optical field and accounting for the highly stochastic detection bottleneck at ultra-low optical energy levels, our approach can enhance image sensing accuracy with minimal detection energy required.

In the final chapter, we summarize potential future directions based on the work presented in this thesis.

CHAPTER 2

BACKGROUND INFORMATION

2.1 Neural-Network Computation

Neural networks, inspired by the biological neural networks that constitute animal brains, are a class of models in machine learning and artificial intelligence. They are designed to recognize patterns and structures in data through a process that mimics the way human brains learn and process information. The core component of neural networks is the layer, each consisting of a linear transformation followed by a non-linear activation function.

2.1.1 Layer Structure

Consider a neural network that takes an input vector $\mathbf{x} \in \mathbb{R}^n$. The transformation in the first layer of the network can be expressed mathematically as follows:

$$\mathbf{a}^{(1)} = \sigma\left(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) \tag{2.1}$$

For each subsequent layer k (where $2 \leq k \leq L$), the output of layer k is given by:

$$\mathbf{a}^{(k)} = \sigma\left(W^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{2.2}$$

where:

- $W^{(k)}$ is the weight matrix for layer k,
- $\mathbf{b}^{(k)}$ is the bias vector for layer k,
- $\mathbf{a}^{(k-1)}$ is the activated output from the previous layer $k-1$,
- σ is the activation function.

The final output $\mathbf{y}$ of the network is then:

$$\mathbf{y} = W^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)} \tag{2.3}$$

Activation Functions

Activation functions play a critical role in introducing non-linearity into neural networks, enabling them to learn complex patterns. Common activation functions include:

- **Sigmoid**: The sigmoid function maps the input to a value between 0 and 1.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

- **Hyperbolic Tangent (tanh)**: The tanh function maps the input to a value between -1 and 1.

$$\sigma(z) = \tanh(z) \tag{2.5}$$

- **Rectified Linear Unit (ReLU)**: The ReLU function sets all negative inputs to zero and leaves positive inputs unchanged.

$$\sigma(z) = \max(0, z) \tag{2.6}$$

2.1.2 Training Process

Training a neural network involves adjusting the network's weights and biases to minimize a loss function, which quantifies the error between the network's predictions and the actual target values.

Forward Pass

During the forward pass, the input vector $\mathbf{x}$ is propagated through the network layer by layer. Each layer performs a linear transformation followed by a non-linear activation function, ultimately producing the output vector $\mathbf{y}$. This process is described by the equations outlined in the previous section.

Loss Function

The loss function $L(\mathbf{y}, \mathbf{y}_{\text{true}})$ measures the discrepancy between the predicted output $\mathbf{y}$ and the true output $\mathbf{y}_{\text{true}}$. The choice of loss function depends on the type of problem being solved. Common loss functions include:

- **Mean Squared Error (MSE)**: Typically used for regression problems, MSE is defined as:

$$L(\mathbf{y}, \mathbf{y}_{\text{true}}) = \frac{1}{2}\|\mathbf{y} - \mathbf{y}_{\text{true}}\|^2 \tag{2.7}$$

- **Cross-Entropy Loss**: Commonly used for classification problems, cross-entropy loss is defined as:

$$L(\mathbf{y}, \mathbf{y}_{\text{true}}) = -\sum_i y_{\text{true},i} \log(y_i) \tag{2.8}$$

Backward Pass and Gradient Descent

The backward pass, also known as backpropagation, involves computing the gradients of the loss function with respect to each weight and bias in the network. These gradients indicate the direction and magnitude of change required to minimize the loss function. The process of updating the network parameters using these gradients is known as gradient descent.

Gradient Descent Gradient descent is an iterative optimization algorithm used to minimize the loss function. The key idea is to update the weights and biases in the direction that reduces the loss. The update rules for the weights and biases using gradient descent are given by:

$$W_k \leftarrow W_k - \eta \frac{\partial L}{\partial W_k} \tag{2.9}$$

$$\mathbf{b}_k \leftarrow \mathbf{b}_k - \eta \frac{\partial L}{\partial \mathbf{b}_k} \tag{2.10}$$

where η is the learning rate, a hyperparameter that controls the step size of the updates.

Stochastic Gradient Descent (SGD) Stochastic Gradient Descent (SGD) is a variant of gradient descent where the parameters are updated using a single training example or a small batch of examples, rather than the entire training set. This introduces noise into the updates, which can help the model escape local minima and improve generalization.

The update rules for SGD are:

$$W_k \leftarrow W_k - \eta \frac{\partial L_i}{\partial W_k} \tag{2.11}$$

$$\mathbf{b}_k \leftarrow \mathbf{b}_k - \eta \frac{\partial L_i}{\partial \mathbf{b}_k} \tag{2.12}$$

where L_i is the loss for the i-th training example or batch.

Adam Optimizer The Adam optimizer combines the advantages of two other extensions of SGD: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). Adam computes adaptive learning rates for each parameter by maintaining running averages of the gradients and their second moments. The update rules for Adam are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.13}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.14}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.15}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.16}$$

$$\theta_t \leftarrow \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{2.17}$$

where:

- m_t and v_t are estimates of the first moment (mean) and second moment (uncentered variance) of the gradients, respectively,
- β_1 and β_2 are hyperparameters that control the decay rates of these moment estimates,
- g_t is the gradient at time step t,
- ϵ is a small constant to prevent division by zero,
- $\hat{m}_t$ and $\hat{v}_t$ are bias-corrected estimates.

Complete Training Process

The complete training process of a neural network can be summarized in the following steps:

1. **Initialization**: Initialize the weights and biases of the network, typically using small random values.
2. **Forward Pass**: Pass the input data through the network to obtain the output predictions.
3. **Loss Calculation**: Compute the loss function to measure the error between the predictions and the true targets.
4. **Backward Pass**: Perform backpropagation to compute the gradients of the loss with respect to the weights and biases.
5. **Parameter Update**: Update the weights and biases using an optimization algorithm (e.g., SGD, Adam).
6. **Iteration**: Repeat the forward pass, loss calculation, backward pass, and parameter update for a specified number of epochs or until convergence.

During each epoch, the entire training dataset is passed through the network. The process of updating the model parameters for each batch of data within an epoch is known as an iteration. The training process continues for multiple epochs, allowing the network to learn and generalize from the training data.

2.1.3 Inference Process

Inference in a neural network refers to the process of making predictions on new, unseen data using the trained model. This process is crucial for deploying the model in real-world applications. Here, we will detail the inference process, particularly for classification tasks.

Forward Pass for Inference

The inference process involves performing a forward pass through the network with the new input data $\mathbf{x}$. The network computes the output $\mathbf{y}$ by propagating the input through each layer, using the trained weights and biases. Formally, for a neural network with L layers, the forward pass can be described as follows:

Given the input vector $\mathbf{x}$, the output of the first layer is computed as:

$$\mathbf{a}^{(1)} = \sigma\left(W^{(1)}\mathbf{x} + \mathbf{b}^{(1)}\right) \tag{2.18}$$

For each subsequent layer k (where $2 \leq k \leq L$), the output of layer k is given by:

$$\mathbf{a}^{(k)} = \sigma\left(W^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)}\right) \tag{2.19}$$

where:

- $W^{(k)}$ is the weight matrix for layer k,
- $\mathbf{b}^{(k)}$ is the bias vector for layer k,
- $\mathbf{a}^{(k-1)}$ is the activated output from the previous layer $k-1$,
- σ is the activation function.

The final output $\mathbf{y}$ of the network is then:

$$\mathbf{y} = W^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)} \tag{2.20}$$

Classification Tasks

In classification tasks, the goal is to assign an input to one of several predefined classes. The output layer of a neural network designed for classification typically has one neuron per class, and the activation function used in the output layer is often the softmax function. The softmax function converts the output scores into probabilities, which sum to 1, providing a probabilistic interpretation of the classification.

The softmax function is defined as follows:

$$\text{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}} \tag{2.21}$$

where $\mathbf{z}$ is the output vector from the final layer (before applying softmax), and z_i is the i-th element of $\mathbf{z}$.

The class with the highest probability is selected as the predicted class:

$$\hat{y} = \arg\max_i \text{softmax}(\mathbf{z})_i \tag{2.22}$$

where $\hat{y}$ is the predicted class label.

Example of Inference in a Classification Task

Consider a neural network trained to classify images of handwritten digits (0-9). Given a new input image $\mathbf{x}$, the network computes the output vector $\mathbf{y}$ through a series of linear transformations and non-linear activations. After applying the softmax function to the output vector, we obtain a probability distribution over the 10 possible digit classes.

For instance, if the output vector $\mathbf{y}$ after the softmax layer is:

$$\mathbf{y} = [0.1, 0.05, 0.05, 0.1, 0.05, 0.1, 0.1, 0.05, 0.35, 0.05]$$

the predicted class $\hat{y}$ would be 8, as it has the highest probability of 0.35.

2.1.4 Common Architectures

Neural networks come in various architectures, each designed to handle specific types of data and tasks effectively. This section introduces some of the most common neural network architectures: perceptrons, multilayer perceptrons (MLPs), convolutional neural networks (CNNs), and transformers.

Perceptrons

The perceptron is the simplest type of artificial neural network and serves as a building block for more complex networks. It consists of a single neuron with learnable weights and biases. The perceptron applies a step function to the weighted sum of its inputs to produce an output. Mathematically, the perceptron model can be represented as:

$$y = \sigma(\mathbf{w} \cdot \mathbf{x} + b) \tag{2.23}$$

where:

- $\mathbf{w}$ is the weight vector,
- $\mathbf{x}$ is the input vector,
- b is the bias,
- σ is the step function, which outputs 1 if the input is greater than a threshold and 0 otherwise.

Perceptrons are primarily used for binary classification tasks.

Multilayer Perceptrons (MLPs)

A multilayer perceptron (MLP) is an extension of the perceptron that consists of multiple layers of neurons. MLPs include an input layer, one or more hidden layers, and an output layer. Each neuron in a layer is connected to every neuron in the subsequent layer, forming a fully connected network. MLPs use non-linear activation functions, such as the sigmoid or ReLU, to introduce non-linearity into the model, enabling it to learn complex patterns.

Structure

- **Input Layer**: Receives the input data. The number of neurons in this layer corresponds to the dimensionality of the input data.
- **Hidden Layers**: Intermediate layers that apply transformations to the input data. An MLP can have one or more hidden layers, and each hidden layer can have a varying number of neurons.
- **Output Layer**: Produces the final output. The number of neurons in this layer corresponds to the number of classes (for classification tasks) or the number of output variables (for regression tasks).

Forward Pass For an MLP with L layers, the forward pass can be described as follows:

1. Input to the first hidden layer:

$$\mathbf{z}^{(1)} = W^{(1)}\mathbf{x} + \mathbf{b}^{(1)} \tag{2.24}$$

$$\mathbf{a}^{(1)} = \sigma(\mathbf{z}^{(1)}) \tag{2.25}$$

2. For each subsequent hidden layer k (where $2 \leq k < L$):

$$\mathbf{z}^{(k)} = W^{(k)}\mathbf{a}^{(k-1)} + \mathbf{b}^{(k)} \tag{2.26}$$

$$\mathbf{a}^{(k)} = \sigma(\mathbf{z}^{(k)}) \tag{2.27}$$

3. Output layer:

$$\mathbf{z}^{(L)} = W^{(L)}\mathbf{a}^{(L-1)} + \mathbf{b}^{(L)} \tag{2.28}$$

Convolutional Neural Networks (CNNs)

Convolutional neural networks (CNNs) are specialized neural networks designed for processing structured grid data, such as images. CNNs use convolutional layers that apply a set of learnable filters to the input data to extract local features. CNNs are particularly effective for image-related tasks due to their ability to capture spatial hierarchies.

Structure

- **Convolutional Layers**: Apply filters to the input data to produce feature maps. Each filter is convolved with the input to detect specific features.
- **Pooling Layers**: Reduce the spatial dimensions of the feature maps, typically using max pooling or average pooling, to retain the most important information while reducing computational complexity.

- **Fully Connected Layers**: After several convolutional and pooling layers, the output feature maps are flattened and passed through fully connected layers to make the final prediction.

Convolution Operation In a convolutional layer, the convolution operation involves applying a set of filters (or kernels) to the input data to produce feature maps. Each filter is convolved with the input to detect specific features.

Assume the input to the convolutional layer is a 3D tensor $\mathbf{x} \in \mathbb{R}^{H \times W \times C_{\text{in}}}$, where H is the height, W is the width, and C_{in} is the number of input channels (e.g., 3 for an RGB image). Each filter $\mathbf{W} \in \mathbb{R}^{K \times K \times C_{\text{in}}}$ has dimensions $K \times K$ and spans all input channels C_{in}.

The convolution operation for a single filter can be expressed as:

$$\mathbf{z}_{i,j} = \sum_{c=1}^{C_{\text{in}}} \sum_{m=1}^{K} \sum_{n=1}^{K} \mathbf{W}_{m,n,c} \mathbf{x}_{i+m-1, j+n-1, c} + \mathbf{b} \tag{2.29}$$

where:

- $\mathbf{W}$ is the filter,
- $\mathbf{x}$ is the input tensor,
- $\mathbf{z}$ is the output feature map,
- $\mathbf{b}$ is the bias,
- i and j index the spatial dimensions of the output feature map,
- m and n index the spatial dimensions of the filter,
- c indexes the input channels.

The filter slides over the width and height of the input tensor, performing element-wise multiplication and summation (convolution) to produce each element of the output feature map.

If there are C_{out} filters, the output of the convolutional layer will be a 3D tensor $\mathbf{z} \in \mathbb{R}^{H' \times W' \times C_{\text{out}}}$, where H' and W' are the height and width of the output feature maps, and C_{out} is the number of output channels (one for each filter).

Activation Functions CNNs commonly use ReLU as the activation function due to its simplicity and effectiveness in introducing non-linearity.

Pooling Pooling layers down-sample the spatial dimensions of the feature maps. Common pooling operations include:

- **Max Pooling**: Selects the maximum value in each patch of the feature map.
- **Average Pooling**: Computes the average value in each patch of the feature map.

Example Consider a simple CNN for image classification:

1. **Input Layer**: Takes a 32x32 RGB image (3 channels).
2. **First Convolutional Layer**: Applies 16 filters of size 3x3, producing 16 feature maps.
3. **First Pooling Layer**: Applies max pooling with a 2x2 filter, reducing each feature map to 16x16.

4. **Second Convolutional Layer**: Applies 32 filters of size 3x3, producing 32 feature maps.
5. **Second Pooling Layer**: Applies max pooling with a 2x2 filter, reducing each feature map to 8x8.
6. **Fully Connected Layer**: Flattens the feature maps and connects to a fully connected layer with 128 neurons.
7. **Output Layer**: A softmax layer with 10 neurons for classification into 10 classes.

Transformers

Transformers are a type of neural network architecture originally designed for natural language processing tasks. Unlike traditional recurrent neural networks (RNNs), transformers do not rely on sequential data processing and can handle long-range dependencies more effectively. The key innovation in transformers is the self-attention mechanism, which allows the model to weigh the importance of different input tokens dynamically.

Structure A transformer model consists of an encoder and a decoder, each composed of multiple layers of self-attention and feedforward neural networks. The encoder processes the input data, and the decoder generates the output data.

- **Encoder**: The encoder is a stack of N identical layers, each consisting of two main components:
  - Multi-Head Self-Attention Mechanism

  - Position-Wise Fully Connected Feedforward Network

- **Decoder**: The decoder is also a stack of N identical layers with an additional sub-layer for multi-head attention over the encoder's output. Each layer consists of three main components:
  - Masked Multi-Head Self-Attention Mechanism
  - Multi-Head Attention over the Encoder's Output
  - Position-Wise Fully Connected Feedforward Network

Self-Attention Mechanism The self-attention mechanism allows the model to focus on different parts of the input sequence when producing an output. It computes a weighted sum of the input values, where the weights are determined by the similarity between the input values. The self-attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.30}$$

where:

- Q (queries), K (keys), and V (values) are matrices derived from the input.
- d_k is the dimension of the keys.

Multi-Head Attention Multi-head attention enhances the model's ability to focus on different positions. Instead of performing a single attention function, multi-head attention runs multiple attention functions in parallel. The outputs are concatenated and linearly transformed:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)W^O \tag{2.31}$$

where each head is defined as:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.32}$$

Here, W_i^Q, W_i^K, and W_i^V are the projection matrices for the i-th head, and W^O is the output projection matrix.

Position-Wise Feedforward Networks After the multi-head attention layer, a position-wise feedforward network is applied to each position separately and identically. It consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2 \tag{2.33}$$

Positional Encoding Since transformers do not have a built-in notion of sequence order, positional encodings are added to the input embeddings to provide information about the position of each token in the sequence. Positional encodings are added to the input embeddings to retain the positional information:

$$\text{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.34}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d_{model}}}\right) \tag{2.35}$$

where pos is the position, i is the dimension, and d_{model} is the dimensionality of the model.

Applications of Transformers Transformers have been highly successful in various applications, including:

- **Language Translation**: Models like Google's BERT (Bidirectional Encoder Representations from Transformers) and OpenAI's GPT (Generative Pre-trained Transformer) have achieved state-of-the-art results in machine translation tasks.
- **Text Generation**: Models such as GPT-3 can generate coherent and contextually relevant text based on a given prompt.
- **Image Processing**: Vision Transformers (ViTs) have been applied to image classification tasks, leveraging the self-attention mechanism to process images as sequences of patches.

2.2 Optical Neural Networks

Optical neural networks leverage the principles of optics to perform neural network computations. They offer the potential for significant speed and energy efficiency improvements over traditional electronic neural networks.

2.2.1 Basic Principles

Optical neural networks use light to transmit and process information. The typical working mechanism of optical accelerators can be summarized as follows: data, such as matrices and vectors, are encoded in light, utilizing some degree of freedom (e.g., each pixel in a 2D space could represent one element of a vector). This light is then modulated (e.g., by attenuating the light) to implement element-wise products. Then the outputs are focused onto detectors, summing up the element-wise products. In essence, these accelerator systems are like a digital processor's cores (and are especially analogous to matrix compute units found on modern GPUs and accelerators) and process the data in a vectorized fashion, where at each step a batch of products (and the accumulation) happens in parallel. The difference is that the sizes that some ONN platform can process at a time can be significantly larger, such as computing products with vectors of dimension $\geq 10^3$ at a time in some cases. Many ONN platforms share the following typical traits:

Optical Shot Noise Optical systems are subject to errors in both the actual hardware and from photon detection. Detection of optical intensity in particular is subject to a phenomenon known as *shot noise* where the detected value is Poisson distributed: given vectors x and w, with the elements of x encoded as optical intensity, the output Y is distributed as:

$$Y \sim \text{Poisson}(w \cdot x) \tag{2.36}$$

For other encoding schemes such as amplitude or phase encoding, equation

2.36 should be modified, but the detection is still subject to shot noise.

Device Imprecision and Systematic Errors Systematic errors, on the other hand, are not noise but rather errors resulting from deficiencies of the hardware. Unlike noise, systematic errors are identical across multiple attempts to run the same computation. Meanwhile, because data often requires rescaling for input into analog-optical systems, neural networks running optically may encounter scaled errors. Many works studying ONNs have characterized the distribution of errors (prevalence of deviations from ground-truth values) as Gaussian [17, 21].

Free Data Transport and Reuse Transport and copying of data encoded in light is free when performed optically. This negates any cost of having to send data to particular sites to perform computations. Copying may be implemented in a variety of ways, such as via "fanning out" and "fanning in" data (projecting multiple copies, and then focusing multiple computation results onto a detector). However, when splitting a signal in this way, the total amount of light is divided by the number of copies.

Efficient Photon Usage Shot noise, and therefore an optical dot product's signal-to-noise ratio (SNR, which serves as an effective bit precision) is related to the mean number of photons at the *output*. The efficiency of photon usage can therefore grow with increasing multiply-accumulate operations (MACs): the SNR for the product $w \cdot x$ is

$$\text{SNR}(Y) = \frac{\text{E}[Y]}{\sqrt{\text{Var}[Y]}} = \sqrt{w \cdot x} = \sqrt{\text{E}[Y]}, \tag{2.37}$$

which explains this behavior; if the desired output precision does not change, constant photons are required regardless of dot product size. In other words, the amount of optical energy needed is proportional to the number of vector-vector products (due to needing a certain amount of light for each), but not the amount of compute performed. For example, assume the computation of a dimension d vector dot-product between two vectors. If the desired effective precision is roughly 8-bit, then one wishes to detect a maximum of roughly $255^2 = 65025$ photons at the output. If one still requires a $\sim$ 8-bit output with a dot product of size $2d$, only this same number of photons is necessary if the $2d$-sized vectors have similar statistics to the d-sized vectors; each element could be encoded using half the number of photons as before. Work on ONNs has studied this behavior in a variety of scenarios [24, 25, 1, 21].

This efficient scaling is not a guarantee—the required number of photons may be influenced by a model architecture's activation/weight distributions, encoding schemes, precision requirements, etc [47]. Related to the previous example, if the operands of the $2d$-size dot-product have different statistics (ie. the vectors have larger dynamic ranges), or if more precision in the answer is desired for larger dot products, then differing amounts of photons are required for encoding the inputs.

2.2.2 Existing Optical Neural Network Platforms

The key similarity among ONN platforms with these traits is that they can reuse data: they accept a vector as input, but only convert it from digital signal to optical signal once to compute full matrix-vector products in the optical domain (as opposed to reloading the same data from digital-electronic memory repeatedly every time it is needed in the matrix-vector multiplication). While this reuse is achieved in different ways the concept is the same: ONN accelerators can take advantage of free data transport with optics, shot-noise-limited optical energy usage, and methods for reusing optical data to realize an energy-efficiency advantage. Here we list some examples of platforms include (for more details, see Refs. [15, 23]):

- Modulator arrays [48, 49, 17, 50]: Input data is fed into a grid-like structure and routed to rows of resonators, phase-change materials, or similar elements that modulate the light, realizing a matrix-vector multiplication. Data is typically reused via the branching of the waveguides to the rows of modulating elements.
- Mach-Zehnder Interferometer (MZI) meshes [51, 12, 52, 53]: Input data is fed into a cascaded arrangement of MZIs that redistribute optical energy and information, computing a matrix-vector multiplication. The depth of the circuit allows for data flowing through to be reused [54] for all stages of the computation of the matrix-vector product at all depths.
- Spatial-Light-Modulator-based (SLM) ONNs [55, 56, 1]: Data is fanned out, fed through a spatial light modulator that realizes element-wise scalar multiplications, and is then fanned in to compute matrix-vector multipli-

cations. Weights may be kept in place to be used with many input vectors, which are copied via fan out.

- Fourier-domain convolution engines [57]: Input data is fed through passive optical components resulting in the spatial Fourier transform; operations applied in the Fourier domain (such as multiplication by weights) thus correspond to performing a convolution. The application of weights in the Fourier domain is equivalent to reusing a spatial-domain kernel at every pixel of an image.
- Diffractive networks [13, 57, 18, 58, 59]: Inputs pass through a series of diffractive elements, realizing a matrix-vector multiplication. The diffractive elements can distribute the input data in a fashion similar to MZI meshes that leverages optical depth [54], and weights can often be kept in place or are fixed at fabrication time.
- Wavelength-multiplexed vector dot-product engines [16, 21, 60]: Inputs are encoded and modulated as a pulse train through EOMs, realizing vector-vector dot products by collecting element-wise multiplications at detectors. Systems can employ wavelength multiplexing so that multiple data is processed at the same time, allowing for matrix-vector multiplication. Copying data with added delay allows for convolutions [16].

2.3 Gaussian Quantum Optics

Gaussian quantum optics is a fundamental area of quantum optics that deals with quantum states of light whose statistical properties are described by Gaussian functions. It provides a powerful framework for understanding and ma-

nipulating quantum states of light. Its applications in quantum information processing, quantum communication, and quantum metrology highlight its significance in advancing quantum technologies.

2.3.1 Fundamentals of Quantum Optics

Quantum optics is the study of the quantum mechanical properties of light. It involves the quantization of the electromagnetic field and the interaction of light with matter at the quantum level.

Quantization of the Electromagnetic Field

The quantization of the electromagnetic field treats the field as a collection of quantized harmonic oscillators. The electric field operator $\hat{E}(t)$ in one dimension can be expressed as:

$$\hat{E}(t) = i \sum_k \sqrt{\frac{\hbar\omega_k}{2\epsilon_0 V}} \left(\hat{a}_k e^{-i\omega_k t} - \hat{a}_k^\dagger e^{i\omega_k t} \right) \tag{2.38}$$

where:

- $\hat{a}_k$ and $\hat{a}_k^\dagger$ are the annihilation and creation operators, respectively, for the mode k,
- ω_k is the angular frequency of mode k,
- ϵ_0 is the vacuum permittivity,
- V is the quantization volume.

Coherent States

Coherent states are the quantum states of the electromagnetic field that most closely resemble classical light. They are eigenstates of the annihilation operator $\hat{a}$:

$$\hat{a}\,|\alpha\rangle = \alpha\,|\alpha\rangle \tag{2.39}$$

where α is a complex number. Coherent states have a minimum uncertainty product and their quadrature components have equal variances.

Squeezed States

Squeezed states are quantum states of light where the uncertainty in one quadrature component is reduced below the vacuum state level at the expense of increased uncertainty in the conjugate quadrature. A squeezed vacuum state can be represented as:

$$|\xi\rangle = \hat{S}(\xi)\,|0\rangle \tag{2.40}$$

where $\hat{S}(\xi)$ is the squeezing operator defined as:

$$\hat{S}(\xi) = \exp\left(\frac{1}{2}(\xi^{*}\hat{a}^{2} - \xi\hat{a}^{\dagger 2})\right) \tag{2.41}$$

with $\xi = re^{i\theta}$ being the complex squeezing parameter.

Thermal States

Thermal states describe the statistical mixture of number states according to the Bose-Einstein distribution. The density operator for a thermal state is given by:

$$\hat{\rho}_{\text{th}} = \frac{1}{\bar{n}+1}\left(\frac{\bar{n}}{\bar{n}+1}\right)^{\hat{a}^\dagger \hat{a}} \tag{2.42}$$

where $\bar{n}$ is the average photon number.

2.3.2 Gaussian States and Operations

Gaussian states are fully characterized by their first and second statistical moments: the mean value vector and the covariance matrix.

Mean Value Vector and Covariance Matrix

The mean value vector **d** and covariance matrix **V** for a quantum state are defined as:

$$d_i = \langle \hat{R}_i \rangle \tag{2.43}$$

$$V_{ij} = \frac{1}{2}\langle \hat{R}_i \hat{R}_j + \hat{R}_j \hat{R}_i \rangle - \langle \hat{R}_i \rangle \langle \hat{R}_j \rangle \tag{2.44}$$

where $\hat{R} = (\hat{q}, \hat{p})^T$ is the vector of quadrature operators.

Quadrature Basis and $\hat{a}, \hat{a}^\dagger$ Basis

The quadrature operators $\hat{q}$ and $\hat{p}$ are defined as:

$$\hat{q} = \frac{1}{\sqrt{2}}(\hat{a} + \hat{a}^\dagger) \tag{2.45}$$

$$\hat{p} = \frac{1}{i\sqrt{2}}(\hat{a} - \hat{a}^\dagger) \tag{2.46}$$

In the quadrature basis, the annihilation and creation operators $\hat{a}$ and $\hat{a}^\dagger$ can be expressed as:

$$\hat{a} = \frac{1}{\sqrt{2}}(\hat{q} + i\hat{p}) \tag{2.47}$$

$$\hat{a}^\dagger = \frac{1}{\sqrt{2}}(\hat{q} - i\hat{p}) \tag{2.48}$$

These representations are useful for different types of quantum optical calculations. The quadrature basis is often used in the context of Gaussian states and their covariance matrices, while the $\hat{a}, \hat{a}^\dagger$ basis is commonly used in number state and coherent state calculations.

Symplectic Transformations

Symplectic transformations are linear transformations that preserve the commutation relations between quadrature operators. They can be represented by symplectic matrices $\mathbf{S}$ such that:

$$\mathbf{S}\boldsymbol{\Omega}\mathbf{S}^T = \boldsymbol{\Omega} \tag{2.49}$$

where $\boldsymbol{\Omega}$ is the symplectic form matrix. Gaussian operations, such as squeezing, displacement, and beam splitting, correspond to symplectic transformations on the covariance matrix and the mean value vector.

This relation allows us to switch between the two representations depending on the context and the convenience for specific calculations.

Wigner Function

The Wigner function is a quasi-probability distribution function that provides a complete description of the quantum state in phase space. For a Gaussian state, the Wigner function $W(\mathbf{R})$ is given by:

$$W(\mathbf{R}) = \frac{\exp\left(-\frac{1}{2}(\mathbf{R}-\mathbf{d})^T\mathbf{V}^{-1}(\mathbf{R}-\mathbf{d})\right)}{(2\pi)^n\sqrt{\det(\mathbf{V})}} \tag{2.50}$$

where $\mathbf{R}$ is the phase space vector.

CHAPTER 3

A LARGE-SCALE FREE-SPACE OPTICAL MATRIX-VECTOR MULTIPLIER

Much of the content in this chapter is adapted from the work presented in Refs. [1, 4].

3.1 Background

The key to realizing an energy advantage in an optical matrix-vector multiplier is to maximize the sizes of the matrices and vectors that are to be multiplied. With large operands, there are many constituent scalar multiplication and accumulation operations that can be performed in parallel completely in the optical domain, and the costs of conversions between electronic and optical signals can be amortized [25]. Optics provides several different ways to implement operations in parallel, including using wavelength multiplexing [17, 16], spatial multiplexing in photonic integrated circuits [12, 17, 61, 33, 52, 62], and spatial multiplexing in 3D free-space optical processors [13, 63, 64, 65, 66, 57, 67, 68, 56, 69, 18].

To date, across all multiplexing approaches and architectures, demonstrations of analog ONNs have involved small vector-vector dot products (as a fundamental operation in implementing convolutional layers [17, 16] and fully connected layers [56]) or matrix-vector multiplications (for realizing fully connected layers [12]): the vectors have been limited [70] to sizes of at most 64. This is substantially below the scale (vector sizes $>10^3$) at which sub-photon-per-multiplication energy efficiency is predicted to be feasible [24, 25]. This is

the fundamental reason that the optical energy consumption in recently demonstrated optical processors is still several orders of magnitude higher than that of theoretical predictions (10^{-14}-10^{-13} versus 10^{-18} J per scalar multiplication) [24, 25, 16, 56, 70]. One ONN architectural approach that is promising for near-term explorations of large-scale ONN operation is to perform spatial multiplexing in 3D free space, since a 2D cross section can contain [71], for example, $>10^6$ modes in an area of 1 mm^2. While the potential for large-scale operation exists based on the available parallelism in free-space spatial modes, this potential has not yet been realized.

In this chapter, we report on our experimental implementation of a 3D free-space optical processor that can perform matrix-vector multiplications at large scale (with vector sizes of up to 0.5 million). The large scale has enabled us to demonstrate the computation of matrix-vector products and vector-vector dot products each using less than 1 photon per scalar multiplication.

3.2 Overview

Here we demonstrate an optical processor to compute analog matrix-vector multiplications at a scale that permits ultra-low optical energy consumption. Our proof-of-concept processor was realized in free space to achieve large vector size by leveraging the large number of spatial modes for multiplexing. By shaping the vectors and corresponding weights into 2D blocks (Figure 3.1a), our device can compute up to ~ 0.5 million scalar multiplications and additions in a matrix-vector multiplication fully in parallel in a single pass of light through the setup. In the special case where the matrix is a *vector* of size $\sim 5 \times 10^5$ (so

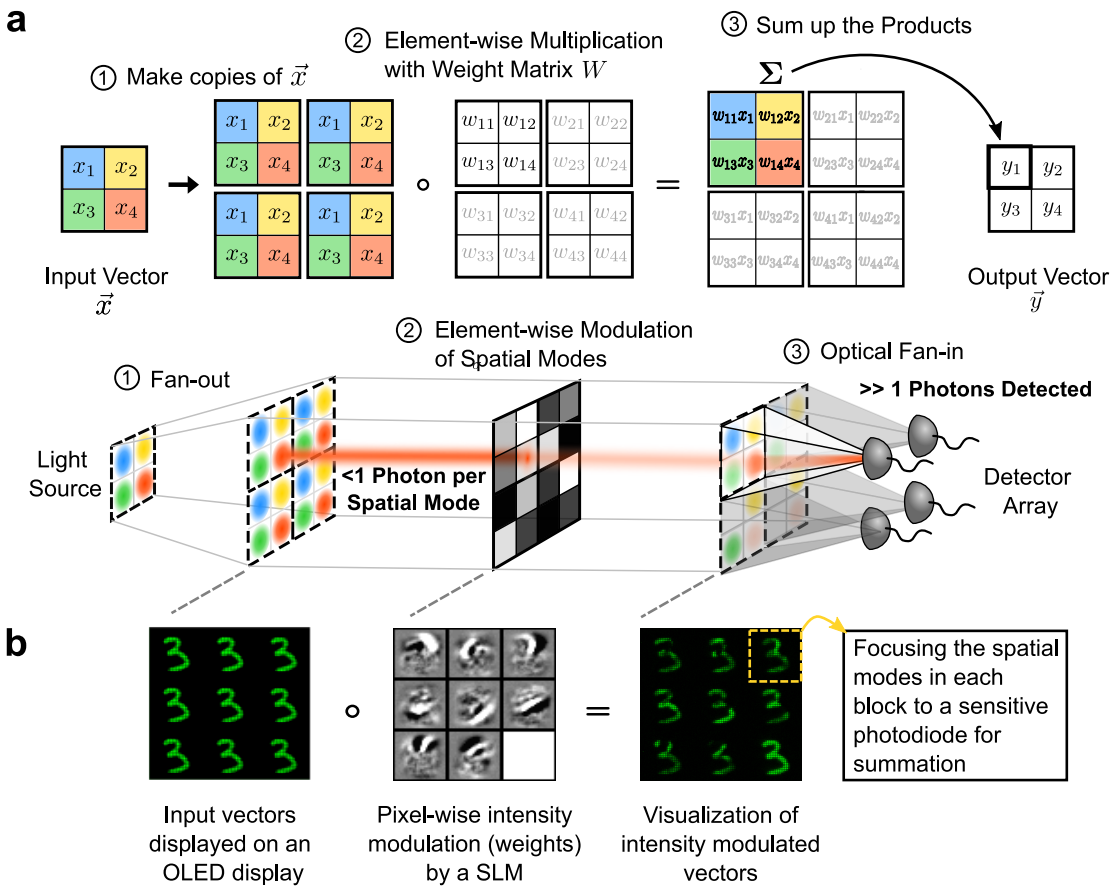


Figure 3.1: **The working principle of the optical matrix-vector multiplier** (a) The schematic of our optical matrix-vector multiplier design (a vector size of 4 is used for illustration purposes). The multiplier computes the matrix-vector product $y_i = \sum_j^4 w_{ij} x_j$ in three steps: the top row illustrates the mathematical operations equivalent to matrix-vector multiplication, and the bottom row shows the corresponding physical implementation of each operation. Each element of the vector $\vec{x}$ is color-coded to show the correspondence between the mathematical and physical operations. OLED: organic light-emitting diode; SLM: spatial light modulator. (b) Visualization of optical computation of matrix-vector multiplication with the image of a handwritten digit as the input vector.

the matrix-vector product is a vector-vector dot product), we obtained an average error of $\sim 6\%$ (or a precision of 4 noise-equivalent bits [25]) in dot-product computation at ~ 0.001 photons per scalar multiplication. Even though our demonstration was made with free-space optics, it was designed to illustrate the key scientific points in Refs. [25, 24]: the energy consumption of optical matrix-vector multiplicaiton can be amortized with large vectors sizes. Our findings are generalizable to a wide variety of both integrated and non-integrated ONN implementations.

3.3 Experimental Setup

The optical vector-vector dot product multiplier setup consists of an array of light sources, a zoom lens imaging system, an intensity modulator, and a photodetector (Figure 3.2). We used an organic light-emitting diode (OLED) display of a commercial smartphone (Google Pixel 2016 version) as the light source for encoding input vectors. The OLED display consist of a 1920×1080 pixel array, with individually controllable intensity for each pixel (for details, see 3.3.1). A reflective liquid-crystal spatial light modulator (SLM, P1920-500-1100-HDMI, Meadowlark Optics) was combined with a half-wave plate (HWP, WPH10ME-532, Thorlabs) and a polarizing beamsplitter (PBS, CCM1-PBS251, Thorlabs) to perform intensity modulation as weight multiplication (for details, see 3.3.2). The SLM has a pixel array of dimensions 1920×1152, with individually controllable transmission for each pixel. A zoom lens system (Resolv4K, Navitar) was used to image the OLED display onto the SLM panel (for details, see 3.4.1). The intensity-modulated light field reflected from the SLM was further de-magnified and imaged onto the detector, by a telescope

formed by the rear adapter of the zoom lens (1-81102, Navitar) and an objective lens (XLFLUOR4x/340, Olympus). An additional band-pass filter (BPF, FF01-525/15-25, Semrock) and polarizer (LPVISE100-A, Thorlabs) were inserted into the telescope (Figure 3.2) in order to reduce the bandwidth and purify the polarization of the light reflected by the PBS, resulting in more precise results. During alignment and troubleshooting, we used a camera (Prime 95B Scientific CMOS Camera, Teledyne Photometrics) as a multi-pixel detector (Figure 3.3a). For sensitive measurements under extremely low photon fluxes, we used a multi-pixel photon counter (MPPC, C13366 series GA type, Hamamatsu Photonics) as a bucket detector (Figure 3.3b) (for details, see 3.6). When it was necessary to further reduce the optical power, an additional neutral density filter (ND=0.4, NE2R04B, Thorlabs) was placed in front of the zoom lens to attenuate light.

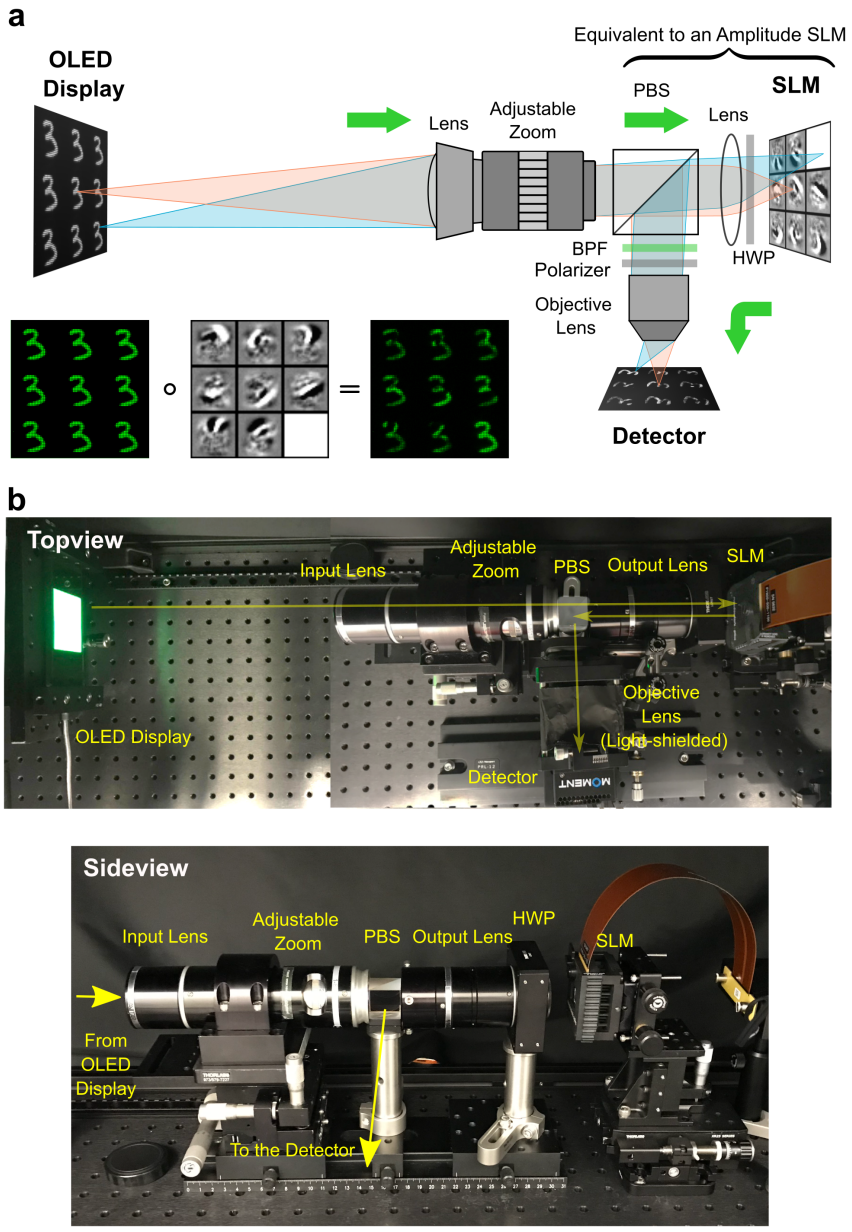


Figure 3.2: **Experimental layout for the optical vector-vector dot product multiplier setup. a,** The schematic of the optical setup. An illustration of the element-wise multiplication is also shown. In this example, nine copies of an input vector of handwritten digits, all '3'—which our setup accepts as 2D images—are intensity modulated by different weight vectors, which are each encoded as a 2D block on the spatial light modulator. Images of the vectors before and after the intensity modulation were taken by a camera placed at the detector location. (PBS: polarizing beam splitter; HWP: half-wave plate; BPF: band-pass filter; SLM: spatial light modulator) **b,** Photos of the core setup corresponding to the schematic are shown in panel (a).

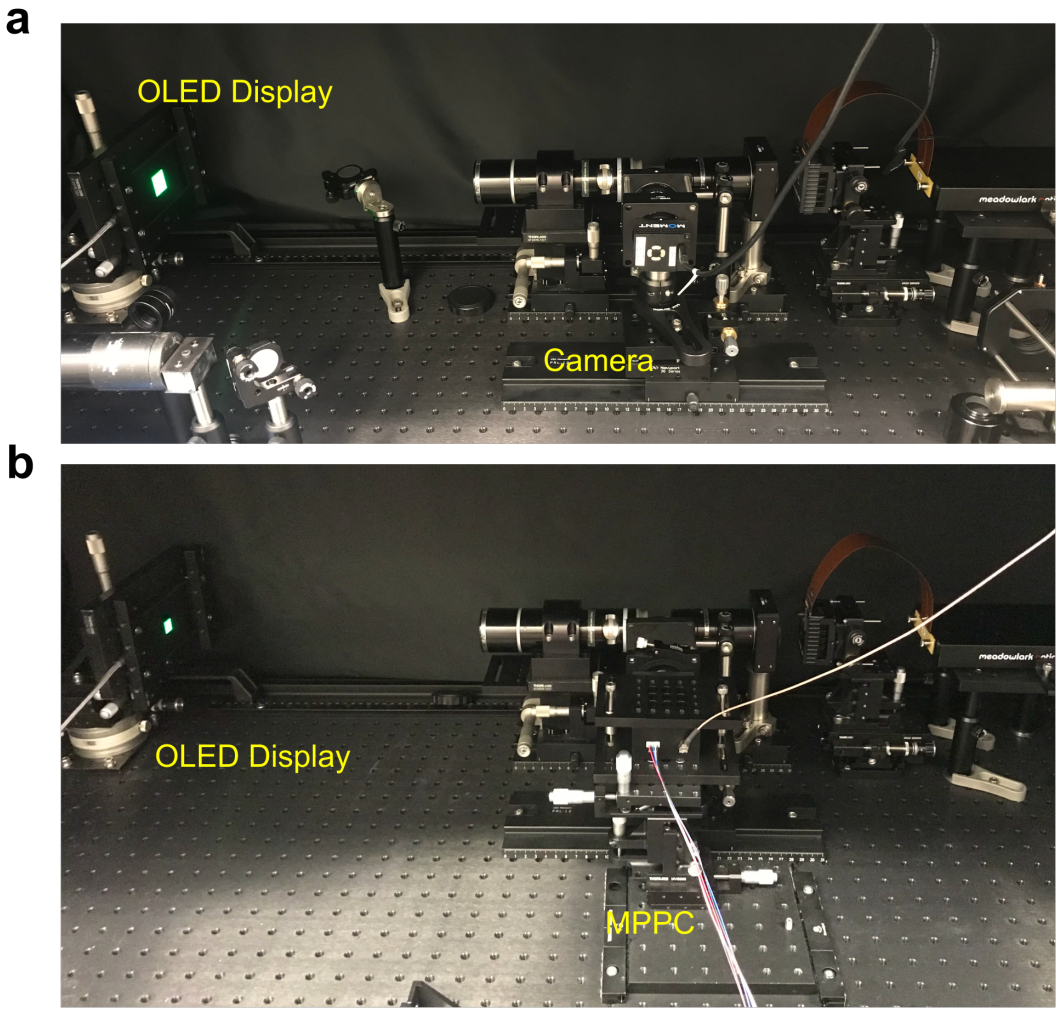


Figure 3.3: **Photos of the entire setup.** **a,** The setup configured with a CMOS camera as a multi-pixel detector. **b,** The setup configured with an MPPC as a sensitive bucket detector.

3.3.1 Properties of the OLED Display

We chose an OLED display, which is made up of spatially and temporally incoherent light sources, to encode input vector $\vec{x}$ for several reasons. OLED pixels feature a high extinction ratio and high dynamic range in intensity, which are ideal for characterizing the accuracy of vector-vector dot products. A commercial-grade integrated OLED panel with a high pixel count is readily available at a low cost, which made it possible to encode very large vectors that were essential for demonstrating vector-vector dot products on our setup. Even though OLEDs are unable to encode phase information like coherent sources (e.g., lasers), our setup design based on optical imaging is compatible with both coherent and incoherent light.

Compared to other options of integrated incoherent light sources (e.g., liquid-crystal display, LCD), OLED pixels can be turned off completely, while LCD screens are always backlit with LED panels and thus always transmit some residual light. The true darkness of OLED pixels allowed us to achieve high dynamic range in intensity modulation and to reduce noise caused by background light pollution. Finally, the OLED display pixels used in this experiment were conveniently point-shaped and arranged in the same square lattice array as the SLM pixels (Figure 3.4a), which facilitated pixel-to-pixel alignment (see 3.4.1), and enabled us to unambiguously quantify the number of spatial modes, each performing one analog scalar multiplication. In contrast, commercial LCD pixels are typically arranged in bars, which require optical image transformation before they can be aligned to SLM pixels.

The OLED display used in our study had three different colors of pixels: red, blue, and green. We used only the green pixels, which form a 1080×1920 square

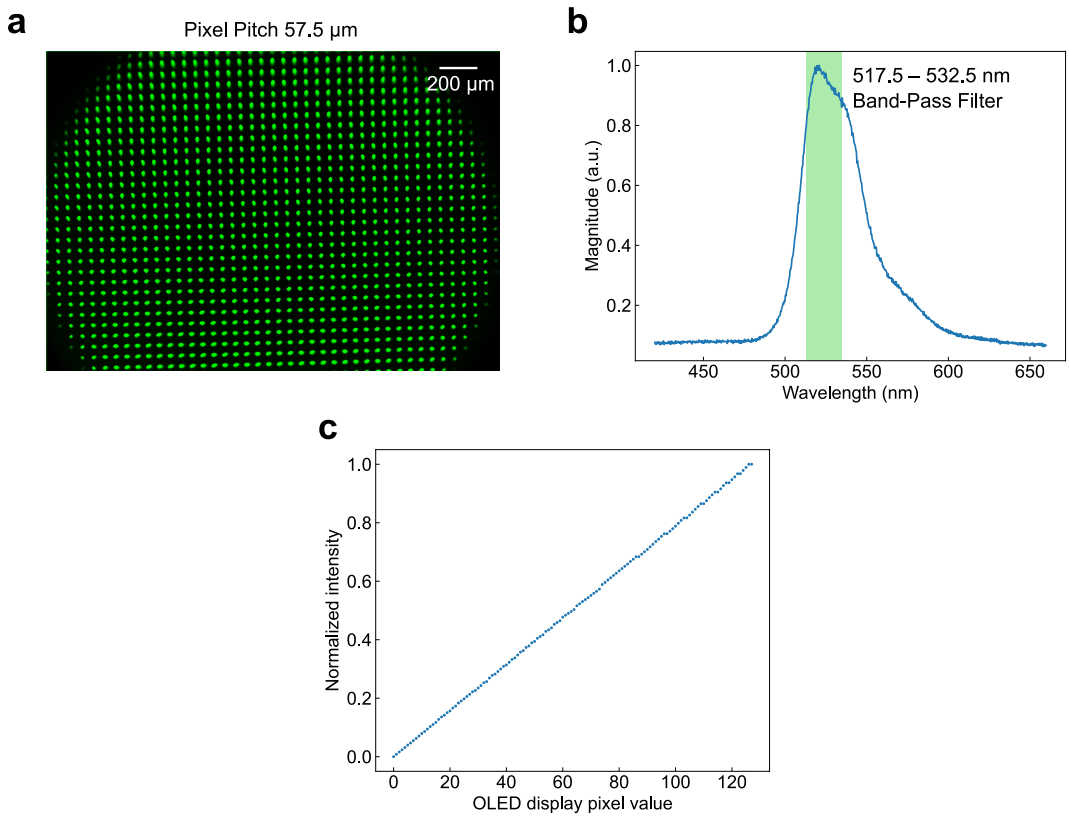


Figure 3.4: **Properties of the organic light-emitting diode (OLED) display. a,** An image of the green OLED pixels taken under an inspection microscope. The pixels form a square lattice with a pixel pitch of 57.5 µm. Scale bar, 200 µm. **b,** Emission spectrum of the green OLED pixels. The shaded area indicates the transmission band of the BPF (> 90% transmission). **c,** The 7-bit linear look-up table (LUT) calibrated to control the OLED display intensity.

lattice array (~2×10^6 total pixels) as shown in Figure 3.4a. The pixel pitch was measured to be 57.5 µm. The maximum power of each pixel was measured to be ~1 nW, emitted in a very wide angle (> 60 degrees). Since the light emitted from the OLED screen had a rather broad spectrum, we used a band-pass filter (FF01-525/15-25, Semrock) to reduce the bandwidth in order to improve coherence for more precise and stable phase modulation by the SLM (Figure 3.4b). The intensity of each individual pixel can be controlled independently with 256

(8-bit) control levels. However, since the actual output intensity was not linear with the pixel control level, we calibrated a linear look-up table (LUT) that contains 124 distinct intensity levels (~7 bits, Figure 3.4c).

3.3.2 Intensity Modulation with a Phase-Only SLM

We converted a phase-only SLM into an intensity modulator with a half-wave plate (HWP) and a polarizing beam splitter (PBS). The SLM pixels are made of birefringent liquid crystal layers, whose refractive index can be tuned by applying voltage across them. By controlling the refractive index of extraordinary light, the SLM pixels introduce a phase difference $\phi_e - \phi_o$ between the extraordinary and ordinary light, whose polarizations are perpendicular to each other. When a PBS and HWP were placed in front of a reflective SLM, the light field passed the components twice, once during the trip towards the SLM and once after being reflected by the SLM (Figure 3.2a). One of the functions of PBS was to separate the output from the input light: the input light (incident to the SLM) was horizontally polarized and transmitted by the PBS, while the output light (reflected from the SLM) was vertically polarized, and therefore reflected by the PBS. The other function of the PBS is to convert the polarization state of the output light to its amplitude: the light modulated by the SLM was in general elliptically polarized, controlled by the phase difference $\phi_e - \phi_o$. The amplitude of the light field (and intensity in this case too) was modulated by selecting only the vertical component of the SLM-modulated light at the output port of the PBS. The HWP was placed with its fast axis rotated 22.5 degrees from the extraordinary axis of the SLM such that the intensity transmission could be tuned from 0 to 100%. Figure 3.5a shows the calculated relationship between the inten-

sity transmission and phase difference $\phi_e - \phi_o$. The maximum extinction ratio of the transmission intensity was measured to be ~50 (Figure 3.5b). The SLM consists of $1920 \times 1152 \sim 2.2 \times 10^6$ pixels, each of which can be independently controlled for intensity modulation with a 256 (8-bit) LUT (Figure 3.5c). Alternatively, instead of using a phase-modulation SLM, the intensity modulator can be more compactly implemented with a monolithic LCD panel in a transmission geometry.

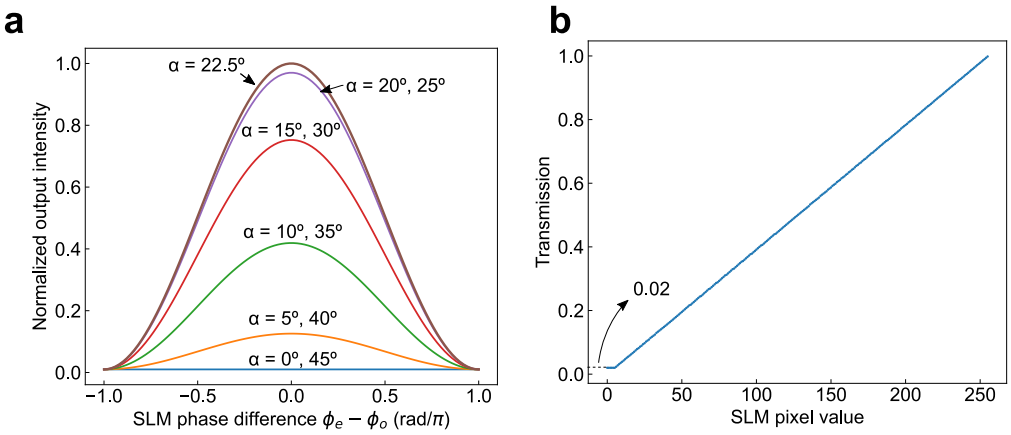


Figure 3.5: **Intensity modulation with a spatial light modulator (SLM).** **a,** Simulation results of output intensity as a function of phase difference $\phi_e - \phi_o$. α is the angle between the half-wave plate (HWP) fast axis and the extraordinary axis of the SLM. **b,** The 8-bit LUT of the SLM for intensity modulation. The minimum transmission was measured to be ~0.02 times the maximum transmission, which is equivalent to an extinction ratio of ~50.

3.3.3 Characterization of the Photodetector

For single-photon detection, we used a multi-pixel photon counter (MPPC) as a bucket detector. We chose an MPPC for its high signal-to-noise ratio (SNR), large measurement range, and moderately high bandwidth. The MPPC is composed of an array of Geiger-mode photodiodes with high intrinsic gain, which

enables the photodiodes to detect single-photon events. The detection of each photon results in a spike-shaped impulse response in the output voltage of the detector, with a sharp rising edge and an approximately exponentially decaying tail (Figure 3.6a).

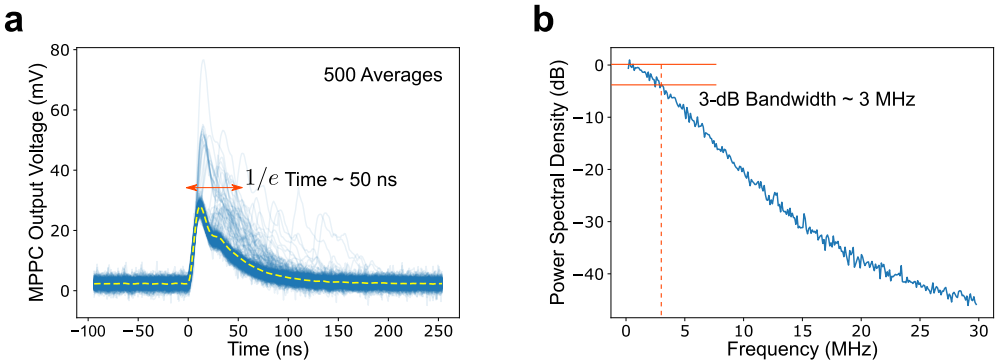


Figure 3.6: **Time and frequency characteristics of the multi-pixel photon counter (MPPC). a,** The impulse response of single-photon detection averaged over 500 trials. **b,** The frequency response and Power Spectral Density of the MPPC.

When the photon flux rate is extremely low ($<10^7$ photons per second), the detected photons can be enumerated by counting the number of spikes (Figure 3.7a). The maximum measurable photon flux rate is limited by the bandwidth of the detector (Figure 3.6b) and potentially the dead time after each photon detection. To increase the maximum measurable photon flux (or optical power), the MPPC detector was spatially multiplexed with a 60×60 photodiode array (with 50 µm×50 µm pixel pitch), the outputs of which are then pooled into a single analog voltage trace. When the photon flux far exceeds the MPPC bandwidth ($\gg 10^7$ photons per second), the pulses induced by individual photons overlap in time and can no longer be resolved (Figure 3.7b). In this scenario, we measured the average output voltage, which maintains an excellent linear relationship with the average optical power impinging on the detector. The MPPC output voltage was calibrated against the power reading of a semiconductor power

meter (818-UV-L-FC/DB, Newport), and the calibration result closely agreed with the manufacturer's specifications of the MPPC (Figure 3.7d). Therefore, we were able to use the detector as a fast power meter to measure instantaneous optical power from pW up to several nW (Figure 3.7d). In this experiment, optical power ¿6 pW was measured by converting the output voltage of the MPPC to the optical power impinging on the detector. Compared to regular semiconductor power meters without intrinsic gain—which can also measure ~pW levels of optical power—the MPPC can maintain a high SNR for a much higher bandwidth (~3 MHz, Figure 3.6b), since its signal is amplified to overcome the noise integrated over the larger bandwidth.

When the MPPC is used as a power meter, the minimum power that can be measured is determined by the analog noise floor (including dark counts, thermal noise, and other electronic noises), which was measured to be equivalent to 1.25 pW optical power at the full bandwidth (Figure 3.7c). The dark count of the MPPC was measured to be ~10^4 photons per second (¡10 fW), which accounted for less than 1% of the total noise. Therefore, the detector can in principle measure optical power even below the analog noise equivalent power of 1.25 pW by means of photon counting. In our experiments, the photon counting measurement was conducted only to verify that the detector could indeed resolve single-photon events, and to determine the minimum valid optical power it could measure (~10 fW). Since the optical powers involved in our experiments were higher than the analog noise floor (~1.25 pW), they were all measured via the direct readout of detector's output voltage.

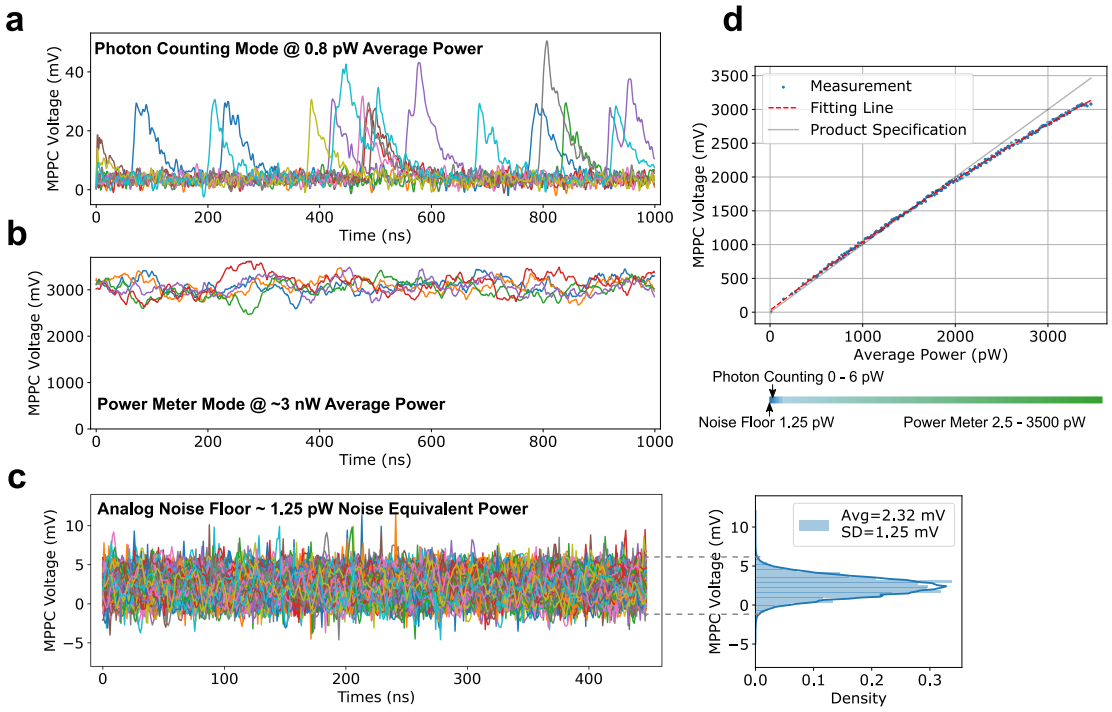


Figure 3.7: **Photon detection and optical power measurement with the MPPC. a,** Single-photon detection under low photon flux. Different colors indicate instances of independent measurement trials. **b,** Instantaneous optical power measurement under high photon flux. **c,** The detector noise floor, which determines the lowest measurable optical power when the MPPC is used as a power meter. The left panel shows examples of independent measurements of the baseline analog noise; the right panel shows the noise distribution and statistics. **d,** The linear relationship between the average MPPC output voltage and the average optical power impinging on the detector. Calibration of the optical power (plot group "Measurement") was performed independently with a semiconductor power meter.

3.4 System Calibration

3.4.1 Alignment of the Optical Imaging System

In order to maximize the vector-vector dot product multiplication size—and thus maximize the energy benefits of optical processing—we aligned as many pixels as possible from the OLED display to the SLM. Three conditions must be satisfied for this pixel-to-pixel alignment:

1. The OLED display must be imaged onto the SLM with a precise de-magnification factor to match the pitch of OLED pixels to that of the SLM pixels.
2. The imaging resolution needs to be high enough such that the image of each OLED pixel on the SLM must be no larger than the size of an SLM pixel. This is to prevent crosstalk.
3. The image of each OLED pixel must be aligned to the corresponding pixel on the SLM, which requires fine adjustment of the translation, rotation, pitch, and yaw of each device involved.

To match the pixel size of the OLED pixel image to the SLM pixel size, the zoom lens was set at a de-magnification of $9.2\,\mu m/57.5\,\mu m = 0.16$. This zoom factor was achieved when the zoom lens (Resolv4K 1-80100, Navitar) was configured with a 0.25× lens attachment (1-81201, Navitar) and 1× rear adapter (1-81102, Navitar). The zoom factor of the zoom lens could be mechanically tuned continuously to precisely match the OLED and SLM pixel pitch. Under this configuration, the spot size in the object plane of the zoom lens (the OLED side) is

40.85 µm in diameter (Rayleigh criterion) according to the manufacturer's specifications. This spot size is smaller than the OLED pixel pitch size of 57.5 µm. Meanwhile, the spot size in the imaging plane (the SLM side) is specified to be 6.52 µm in diameter (Rayleigh criterion), which is also smaller than the SLM pixel pitch size of 9.2 µm. In fact, the performance of the zoom lens system was close to the diffraction limit, and the images of OLED pixels on the SLM plane were well separated (Figure 3.8). Therefore, the setup achieved the correct demagnification factor and possessed adequate resolution, and both conditions (1) and (2) were satisfied.

To align each OLED pixel to the corresponding SLM pixel, mechanical alignment was performed using the following method (Figure 3.8). First, identical images of the same size were displayed on both the OLED display and SLM. The bright pixels on the OLED display corresponded to pixels of full transmission on the SLM. The dark pixels on the OLED display corresponded to the pixels of zero transmission on the SLM. Therefore, the SLM functioned like a mask, with its light-transmitting parts identical in shape and size to the bright image on the OLED display. After intensity modulation by the SLM, the image on the OLED can only be preserved without any clipping if and only if the OLED and SLM pixels are exactly aligned to each other, and with the correct orientation (Figure 3.8).

The maximum number of pixels that could be aligned was determined by the imaging error on the side of the field-of-view (FOV) of the zoom lens. According to the specifications of the zoom lens, at most 3.6 million OLED pixels in a square array can be aligned to the SLM. However, this estimation assumes diffraction-limited performance across the entire FOV. In practice, we managed

to align $711 \times 711 \sim 0.5$ million pixels. There were three reasons for the deterioration of pixel-to-pixel alignment towards the side of the FOV:

1. Vignette: The optical transmission drops off towards the edge of the FOV, which causes up to a ~90 % decrease in intensity for the 711×711 pixel array. As a result, the pixels around the center of the FOV must be dimmed in order to keep all pixels at the same brightness, as required for the computation of large dot products.
2. Image Distortion: nonlinear image distortion (such as barrel, or pin cushion, or other higher-order distortions) lead to a non-uniform local zoom factor of the OLED pixel image. Although linear image distortions can be corrected by mechanical alignment, nonlinear distortions cannot be completely fixed by alignment. Even slight distortions can cause pixels to slowly drift away from each other until eventually there is misalignment towards the side of the FOV.
3. Aberration: The aberration of the zoom lens increases towards the edge of its FOV and causes the focal spots to deviate from the diffraction-limited spot size. This causes the expansion of the OLED pixel image on the SLM plane, which couples part of the optical energy emitted from each OLED pixel into the surrounding SLM pixels that have incorrect transmissions for weight encoding.

The non-uniform transmission caused by the vignette could be fixed by making a pixel-wise LUT of the OLED display, which is described in 3.4.2. While there is no easy solution to nonlinear image distortion and aberration, we characterize them in 3.4.3.

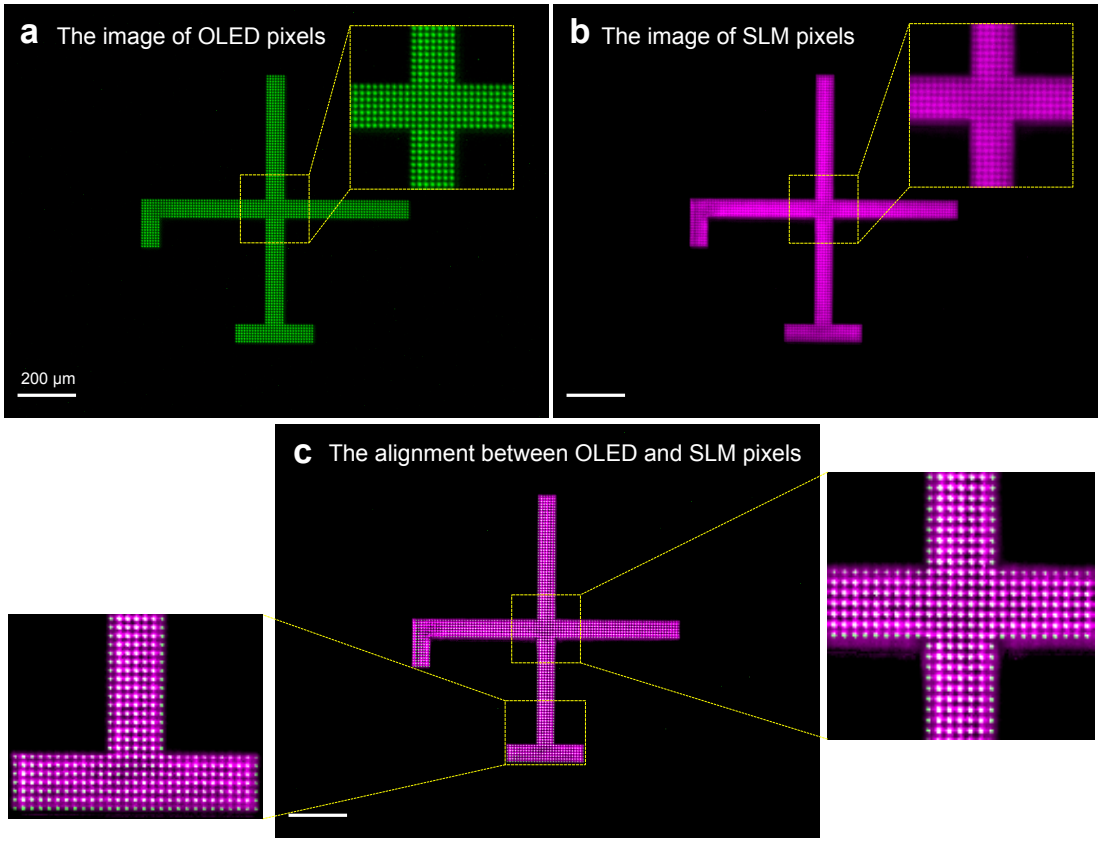


Figure 3.8: **Pixel-to-pixel alignment between the OLED and SLM pixels. a,** The image of a viewfinder pattern on the OLED display. The scale bar measures the distance on the SLM panel, 200 µm. **b,** The image of the identical viewfinder pattern modulated by the SLM. The entire SLM panel was uniformly illuminated by an ambient light source. The SLM performed intensity modulation which functioned as a mask with a hollow of the viewfinder shape. **c,** Visualization of the alignment between the OLED and SLM pixels.

3.4.2 Correction of Optical Vignette

To enable computation of large dot products, we corrected for intensity fall-off towards the edge of the FOV, caused by optical system vignettes as discussed in 3.4.1. The correction was especially important for optical fan-in (discussed in

3.6), since each pixel, regardless of its position in the FOV, should contribute the same amount of optical energy to the detector, if set at the same pixel value.

Correction was performed by making an attenuation map that compensates for different transmissions of pixels at different locations. We first configured the OLED to display at the maximum pixel brightness uniformly across the entire FOV, and then captured an image of the OLED display at the detector plane. Due to the vignette effect of the optical system, the intensity distribution was not uniform at the detector plane (Figure 3.9, top left panel). A region of interest (ROI) was then chosen, in which we sought to achieve uniform intensity for all OLED pixels. The minimum intensity in the ROI was set as the target intensity value (circled areas in the top left panel of Figure 3.9). The brightness of other OLED pixels was reduced iteratively until the intensity of their images matched that of the target value. The result of the correction is shown in Figure 3.9, top right panel, where uniform intensity was achieved in an ROI of size 720×720. Meanwhile, an attenuation map was established to determine the percentage by which each OLED pixel should be attenuated in order to achieve a uniform output intensity.

3.4.3 Pixel Walk-Off and Crosstalk due to Imaging Imperfections

We examined two aspects of the optical system's imaging quality: walk-off of pixel alignment due to image distortion, and degradation of focal spots due to aberration (both discussed in 3.4.1). To visualize these effects, we captured an image of a sparse 2D grid of pixels displayed on the OLED screen. The grid

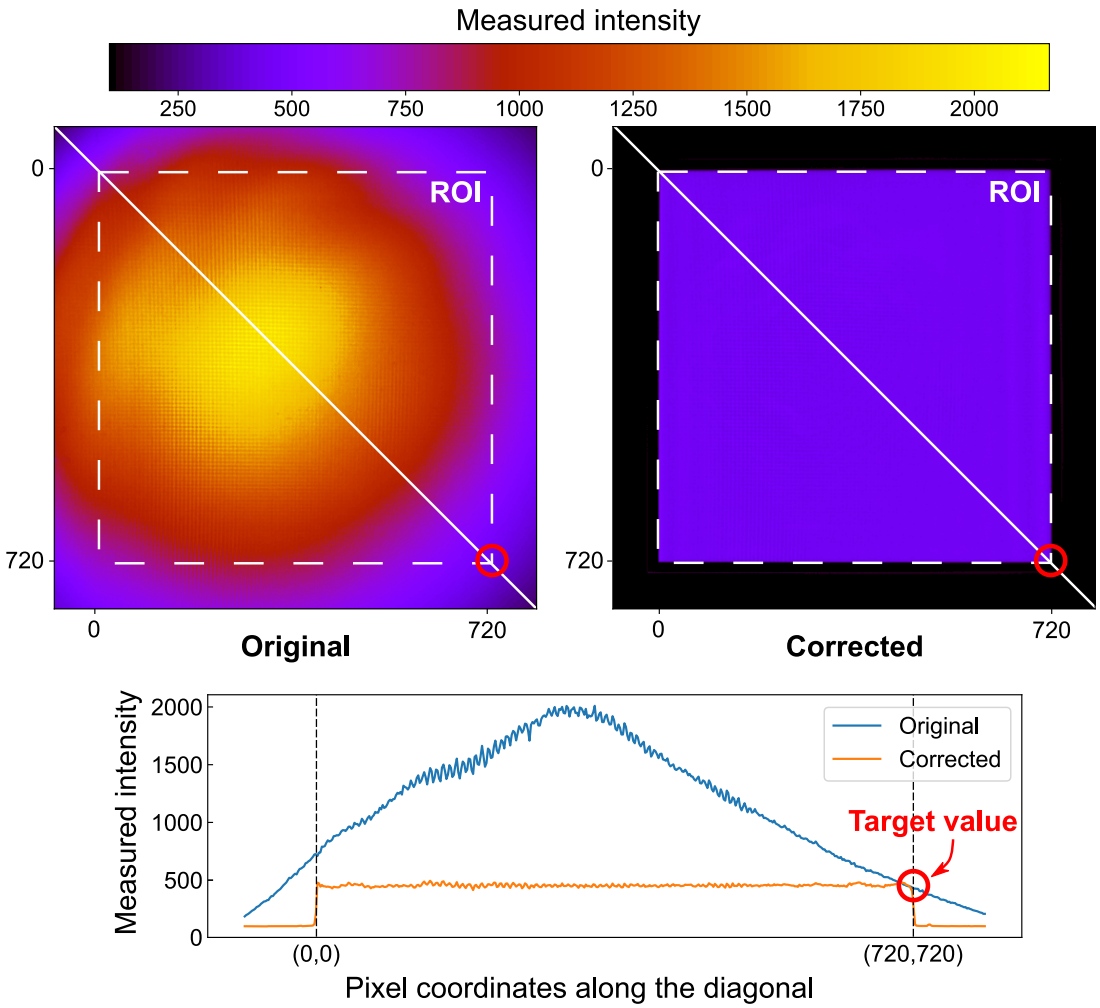


Figure 3.9: **Correction of Non-uniform Transmission of the Optical System.** The original intensity distribution is shown in the top left panel ("Original"), with intensity falling off towards the edges of the region of interest (ROI). The correction procedure reduced the intensity of pixels near the center to match the target value near the darkest corner (720, 720) in the selected ROI. The image after correction is shown in the top right panel ("Corrected"). The bottom panel shows intensity along the diagonal pixels of the ROI (solid white lines) before and after correction.

was composed of blocks with an edge length of 80 pixels, with only the pixel at the center of each block was turned on. Figure 3.10a shows how the imaging quality of single pixels changed across an ROI of 720×720 pixels. For example, images of a single pixel tended to be sharp and focused near the center of the

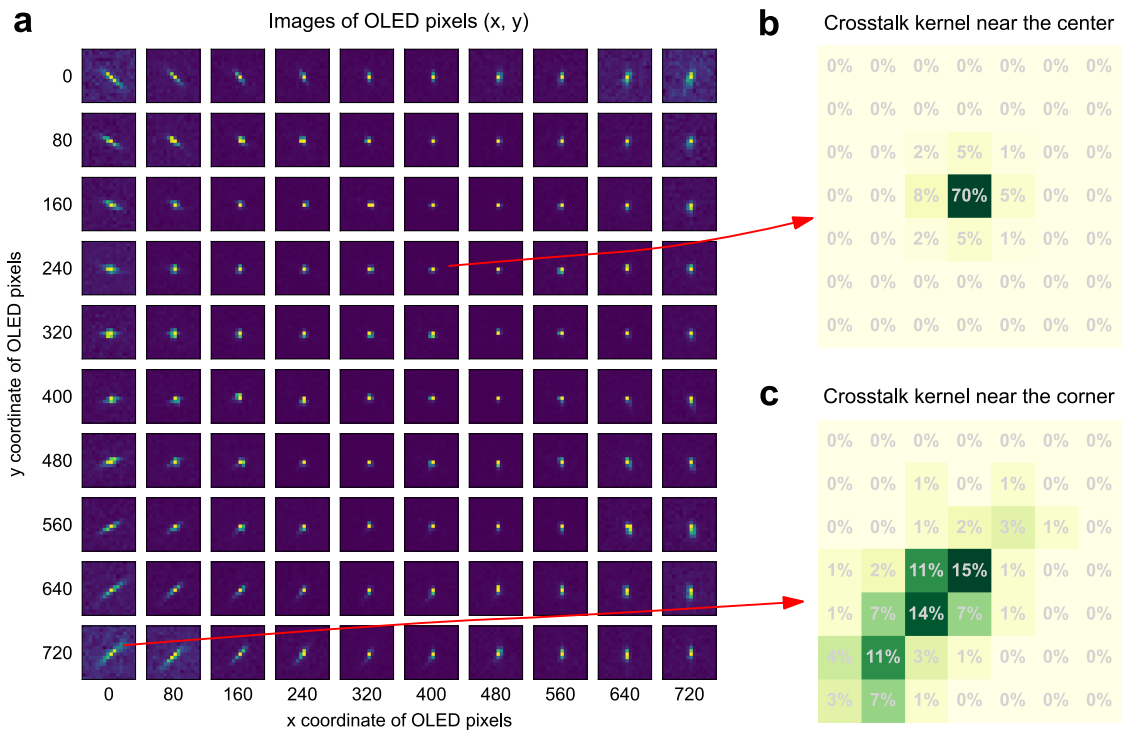


Figure 3.10: **Imaging quality of the OLED pixels. a,** Images of the OLED pixels in a region of interest (ROI) of size 720 × 720. Only the pixels on the grid points of a 2D grid (with edge length of 80 pixels) were switched on. The OLED display was first imaged onto the SLM, and then relayed to the detector plane where the images were captured by a camera. The intensity inside each block was normalized to the maximum pixel value in the block. **b(c),** The 7 × 7 crosstalk kernels near the center (corner) of the FOV. Each entry denotes the percentage of optical power coupled into each individual SLM pixel, with all the power supposed to couple to the central pixel.

FOV, while images of pixels towards the corner of the FOV spread out into a streak along the radial direction. This is probably due to coma aberration, which is common to most imaging systems (Figure 3.10a). Meanwhile, the walk-off of pixel alignment could be observed by the deviation of the focus from the center of each block (Figure 3.10a). As discussed in 3.4.1, pixel walk-off due to linear image distortion can be corrected with careful mechanical alignment,

while nonlinear distortion cannot be eliminated. Based on Figure 3.10a, the pixel walk-off was insignificant for the ROI of 720×720. Incidentally, the largest possible ROI for optical matrix-vector multiplication was determined by the trade-off between optical power transmission and imaging quality distribution (Figure 3.9 and Figure 3.10a). Since the imaging quality was better on the right side, the ROI was shifted slightly to the right of the brightest part of the intensity distribution.

Pixel walk-off and crosstalk both resulted in errors in weight modulation, due to the coupling of optical energy into neighboring SLM pixels with incorrect modulation weights. These effects were quantified and modeled as so-called crosstalk kernels, which are similar to convolution kernels but vary gradually in space. Figure 3.10b, c show the intensity distribution of a focal spot near the center (corner) of the FOV in a 7×7 block of SLM pixels, with the central pixel denoting the SLM pixel that the bright OLED pixel should align to. When both input vectors and weights were natural images (whose pixel value variation was smoother, and usually constitute the first layer of neural networks for image classification), the error caused by walk-off and crosstalk was less severe. For applications in optical neural networks (ONNs), such imaging errors were modeled during the training process by random affine transforms and 2D convolution to enhance the model's resilience to imaging errors (4.4).

3.4.4 System Noise Characteristics

We examined temporal fluctuations of each part of the system and describe hindrances in approaching shot noise-limited performance. Overall, when the

OLED was set at a constant brightness and the SLM at a constant transmission across all pixels, the SNR of optical power measurements were about half the shot noise-limited SNR (Figure 3.11). Sources of excess noise, in addition to shot noise, include intensity fluctuation of the OLED display, phase instability of the SLM, and the intrinsic noise of the detector. At high optical power, noise from external sources dominates the SNR measurement; as the power decreases, shot noise becomes a dominant source of noise. At extremely low optical power, the intrinsic noise of the detector is mainly responsible for deviations from the shot noise-limited performance.

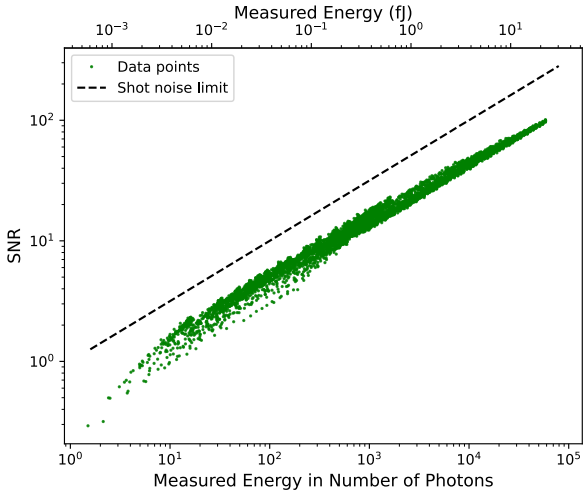


Figure 3.11: **The signal-to-noise ratio (SNR) as a function of photon flux, as measured by the MPPC.** The integration window was 150 ns under analog mode.

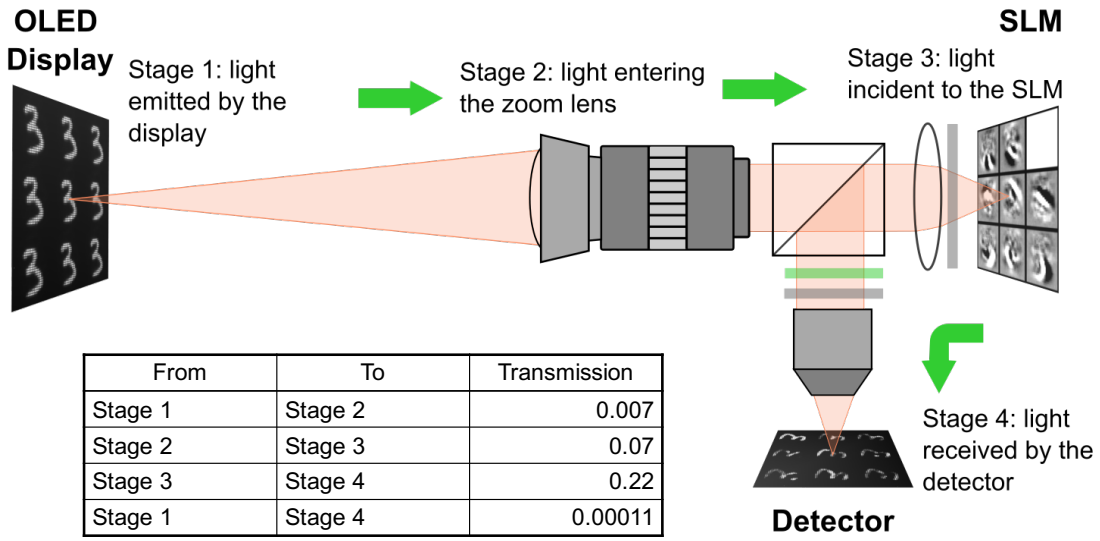
There are three main components causing intensity fluctuations in OLED displays: raster scanning during screen refreshing, pulse width modulation for brightness control, and the thermal noise of OLEDs. When a stationary image was shown on the OLED display, no perceivable raster scanning pattern was observed in high-speed videos of the display. There were occasional black scanning stripes and short bursts of flashing, which are likely to have been caused

by some refreshing mechanism. The OLED display did not seem to use pulse width modulation to adjust brightness until very low brightness settings were reached (below ~35 %). Therefore, we avoided setting pixels to low values, and instead used neutral density filters to attenuate light for extremely low-light measurements. The intensity fluctuation of the light source was mitigated by the high attenuation of the imaging system. Due to the large emission angle of OLED pixels (> 60 degrees), most of the optical power was not collected by the zoom lens, which has a small collection angle. It was estimated that only ~0.7 % of light was collected by the zoom lens (Figure 3.12). The high loss converted the thermal state of the OLED light closer to a coherent state by coupling vacuum states to it [72], which improved the SNR and brought the ratio closer to the shot noise limit. It should be noted that even though the light collection efficiency was low for the zoom lens, the transmission from before the SLM to the detector (where the computation took place) was quite high at ~22 % (Figure 3.12). Thus, optical energy efficiency can be drastically improved if the OLED and zoom lens are replaced with a stable coherent source and an optical system with high transmission in a customized setup.

The phase fluctuation of the SLM stems from the constant switching of voltage across the liquid crystal layers. This fluctuation could be measured by monitoring the intensity fluctuation of the laser diffraction pattern generated by a phase grating on the SLM. According to the manufacturer, the SLM used in this experiment oscillated at 53 kHz, with a measured peak-to-peak power ripple of 0.24% for the first-order diffraction spot. Compared to OLED intensity fluctuations, instabilities caused by the SLM are relatively minor.

The intrinsic noise of the detector (e.g., thermal noise or dark counts) con-

tributed to excess noise that was only apparent when the optical power to be measured was extremely weak. As discussed in 3.3.3, the detector's intrinsic noise was negligible for high optical power ($\gg$1 pW). The analog noise floor becomes significant for low optical power at ~1 pW, which necessitates photon counting for even lower photon flux.

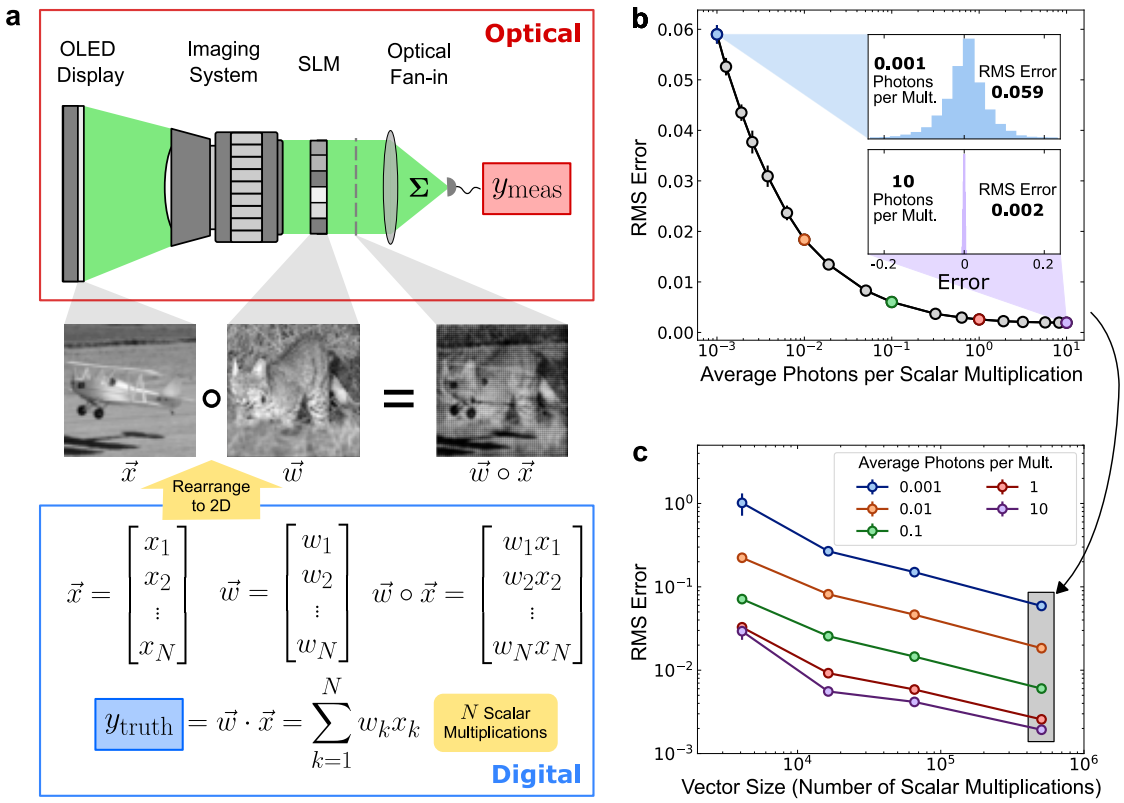


From	To	Transmission
Stage 1	Stage 2	0.007
Stage 2	Stage 3	0.07
Stage 3	Stage 4	0.22
Stage 1	Stage 4	0.00011

Figure 3.12: **The pptical power transmission at each stage of the setup.** A breakdown of the transmission of each stage of the experimental setup is listed in the inset table of the figure.

3.5 Vector-Vector Dot Product Accuracy

To understand how well our system performed in practice in the regime of low optical power consumption, we characterized its accuracy while varying the number of photons used. In our first characterization experiments, we computed the dot products of randomly chosen pairs of vectors (Figure 3.13a; Section 3.5.2). The results from our characterization with dot-product computa-



tions apply directly to the setting of matrix-vector multiplications with generic matrices because our setup computes matrix-vector multiplications as a set of vector-vector dot products. For a dot-product computation, the answer is a scalar, so only a single detector was used: the optical signal encoding the dot-product solution was measured by a sensitive photodetector capable of resolving single photons (Section 3.3.3), and the number of photons used for each dot product was controlled by changing the detector integration time and by inserting neutral-density filters immediately after the OLED display.

To demonstrate our setup could perform computations using less than 1 photon per scalar multiplication for large vector sizes, we measured the numerical precision of dot products between vectors each of size ~0.5 million. With 0.001 photons per scalar multiplication, the error was measured to be ~6% (Figure 3.13b; see Section 3.5.2 for the details of RMS-error calculation); the dom-

Figure 3.13: **Vector-vector dot products computed with high accuracy and high effective numerical precision using as few as 0.001 photons per scalar multiplication. a,** Simplified schematic of optical setup for computation of vector-vector dot products, and characterization procedure. N-pixel images were used as test vectors by interpreting each image as an N-dimensional vector. The setup was used to compute the dot products between many different random pairs of vectors, with each computation producing a result y_{meas} (top and center rows; example experimental measurement of element-wise multiplication $\vec{w} \circ \vec{x}$ was captured with a camera before optical fan-in for illustrative purposes). For each vector pair, the dot-product ground truth y_{truth} was computed on a digital computer (bottom row). The error was calculated as $y_{\text{meas}} - y_{\text{truth}}$. **b,** The root-mean-square (RMS) error of the dot product computation as a function of the average number of detected photons per scalar multiplication. The vector length N was ~0.5 million (711×711), which is sufficiently large that we observed an RMS error of <6% even when only 0.001 photons per multiplication were used, and an RMS error of <1% when 0.1 photons per multiplication were used. The insets show error histograms (over different vector pairs and repeated measurements) from experiments using 10 and 0.001 photons per multiplication, respectively. The error bars in the main plot show 10× the standard deviation of the RMS error, calculated using repeated measurements. **c,** The RMS error as a function of the vector size N, equal to the number of scalar multiplications needed to compute a vector-vector dot product. For each vector size, the RMS error was computed using five different photon budgets, ranging from 0.001 to 10 photons per scalar multiplication. The shaded column indicates data points that are also shown in **b**. The error bars show 3× the standard deviation of the RMS error, calculated using repeated measurements.

inant contribution to this error was from shot noise at the detector (Section 3.4.4). As we increased the number of photons used, the error decreased until it reached a minimum of ~0.2% at 2 photons per multiplication or higher (Figure 3.13b). We hypothesize that the dominant sources of error at high photon counts (>2 photons per multiplication) are imperfect imaging of the OLED display pixels to SLM pixels, and crosstalk between SLM pixels. We note that the experimental runs to test the performance of the system when using 0.001 photons per multiplication (which resulted in ~6% error) were performed with a detector integration time of ~100 ns. This shows that matrix-vector multiplications can be performed with <1 photon per multiplication at a rate of at least 10 MHz, although this is merely an experimentally demonstrated *lower-bound*: in principle, with sufficiently fast modulators and detectors, the system should be able to perform matrix-vector multiplications at rates >10 GHz [25]. To enable comparison between the experimentally achieved analog numerical precision with the numerical precision in digital processors, we can interpret each measured analog error percentage (Figure 3.13b) as corresponding to an effective bit-precision for the computed dot product's answer. Using the metric *noise-equivalent bits* [25], an analog RMS error of 6% corresponds to 4 bits, and 0.2% RMS error corresponds to ~9 bits.

We also verified that we could compute dot products between shorter vectors when using low numbers of photons per scalar multiplication (Figure 3.13c). For photon budgets ranging from 0.001 to 0.1 photons per multiplication, the numerical error was dominated by shot noise for all vector sizes tested. When the number of photons used was sufficiently large, the error was no longer dominated by shot noise, which is consistent with the single-vector-size results shown in Figure 3.13b. For every photon budget tested, dot prod-

ucts between larger vectors had lower error; we attribute this to dot products between larger vectors involving the effective averaging of larger numbers of terms.

3.5.1 Computing Dot Products Using Incoherent Light

Our setup can only perform dot products between vectors with non-negative elements, because x_k is encoded with the intensity of each spatial mode, and w_k is encoded with the transmission of each spatial mode. However, dot products between vectors of signed elements can always be reduced to those between non-negative-valued vectors with minimal digital processing overhead [64]. For a general vector with signed elements $\vec{x}^{\text{signed}}$, where each element $x_k^{\text{signed}} \in [x_{\text{min}}^{\text{signed}}, x_{\text{max}}^{\text{signed}}]$, $x_{\text{min}}^{\text{signed}}, x_{\text{max}}^{\text{signed}} \in \mathbb{R}$, a non-negative vector $\vec{x}'$ can be obtained by adding bias terms and rescaling:

$$\vec{x}' = \frac{x'_{\text{max}} - x'_{\text{min}}}{x_{\text{max}}^{\text{signed}} - x_{\text{min}}^{\text{signed}}} \vec{x}^{\text{signed}} + \frac{x_{\text{max}}^{\text{signed}} x'_{\text{min}} - x_{\text{min}}^{\text{signed}} x'_{\text{max}}}{x_{\text{max}}^{\text{signed}} - x_{\text{min}}^{\text{signed}}} \vec{1}, \tag{3.1}$$

where $\vec{1} = (1, 1, \cdots, 1)$ is a constant vector of size N. It can be verified that x'_k is tightly bounded between $[x'_{\text{min}}, x'_{\text{max}}]$ with $x'_{\text{max}} \geq x'_{\text{min}} \geq 0$. Therefore, two vectors with signed elements $\vec{w}^{\text{signed}}$ and $\vec{x}^{\text{signed}}$ can be converted to non-negative vectors $\vec{w}'$ and $\vec{x}'$, and the dot product $\vec{w}^{\text{signed}} \cdot \vec{x}^{\text{signed}}$ equals to the linear combination of $\vec{w}' \cdot \vec{x}'$, $\vec{w}' \cdot \vec{1}$, $\vec{1} \cdot \vec{x}'$ and a constant term. All three dot products are between vectors of non-negative elements. The dot product between $\vec{1}$ and any vector equals the summation of all of the vector elements, which were computed optically like any other dot product.

In machine learning applications, the input vector $\vec{x}'$ is either the input to the neural network or the neural activation of the previous layer, both of which are non-negative values after the ReLU nonlinear activation function. Therefore, since $\vec{x}^{\text{signed}}$ is already non-negative with $x_{\text{min}}^{\text{signed}} = 0$,

$$\vec{w}^{\,\text{signed}} \cdot \vec{x}^{\,\text{signed}} = c_1 \, \vec{w}' \cdot \vec{x}' + c_2 \, \vec{1} \cdot \vec{x}'. \tag{3.2}$$

For simplicity, both $\vec{x}'$ and $\vec{w}'$ can be normalized to the range $[0, 1]$, and the coefficients in Eq. 3.2 can be solved as: $c_1 = (w_{\text{max}}^{\text{signed}} - w_{\text{min}}^{\text{signed}}) x_{\text{max}}^{\text{signed}}$, $c_2 = w_{\text{min}}^{\text{signed}} x_{\text{max}}^{\text{signed}}$. The normalized vectors $\vec{x}'$ and $\vec{w}'$ were loaded onto the OLED display and the SLM, respectively, according to the hardware LUTs (e.g., Figure 3.4c and Figure 3.5b). In reality, the SLM could not achieve zero transmission which led to a minimum modulation $x_{\text{min}}^{\text{signed}} = \epsilon = 0.02$ (Figure 3.5b). In other words, w_k' was normalized to the range $[\epsilon, 1]$ instead of $[0, 1]$. In this case, $c_1 = \frac{1}{1-\epsilon}(w_{\text{max}}^{\text{signed}} - w_{\text{min}}^{\text{signed}}) x_{\text{max}}^{\text{signed}}$, $c_2 = \frac{1}{1-\epsilon}(w_{\text{min}}^{\text{signed}} - \epsilon w_{\text{max}}^{\text{signed}}) x_{\text{max}}^{\text{signed}}$.

In summary, the dot product between two vectors with signed elements can be converted to two dot products between non-negative vectors, which can be first computed purely with optics, and then combined with only 2 digital multiplications and 2 digital additions. In other words, the price of conversion is a doubling of the amount of optical computation with a constant digital overhead, independent of vector size N.

For matrix-vector multiplication, the computational overhead can be further reduced, since $\vec{1} \cdot \vec{x}'$ remains the same for the dot product between $\vec{x}$ and any row vector of the matrix. As a result, $\vec{1} \cdot \vec{x}'$ only needs to be computed once optically and can be reused afterwards. In other words, to compute a matrix of

size $N' \times N$ multiplied with a vector of size N, in addition to the N' optical dot products (which constitute NN' MACs), only one additional optical dot product ($\vec{1} \cdot \vec{x}$) is required. Thus, the amount of digital overhead is on the order of $O(N')$.

3.5.2 Characterization of Dot Product Accuracy with Varying Photon Budget

Generation of Test Datasets

To generate a test dataset representative of general dot products, we randomly generated vector pairs $\vec{x}$ and $\vec{w}$ based on natural scene images from the STL10 dataset. Each vector was generated from a single color channel of one or more images patched together, depending on the target vector size (each image of size $L \times L$ contributes $N = L^2$ elements to the vector). We chose natural images since they are more representative of the inputs in image classification with globally inhomogeneous and locally smooth features. To adjust the sparsity of the vectors, different thresholds were applied to the image pixel values such that the dot product results cover a wider range of possible values. This was achieved by shifting the original pixel values (float point numbers normalized to the range 0-1) in the entire image up or down by a certain amount, unless the value was already saturated at 1 (the maximum) or 0 (dark). For example, a shift of -1 would make the whole image dark. A shift of +0.2 would make all the pixel values that were originally larger than 0.8 saturated, and would increase all other pixel values by 0.2. This method allowed us to tune the overall intensity of the modulated images without losing the randomness of the distribution.

The computation of dot products on the setup followed the same steps of element-wise multiplication and optical fan-in. Figure 3.14 shows a few more examples of element-wise multiplication, similar to Figure 3.13.

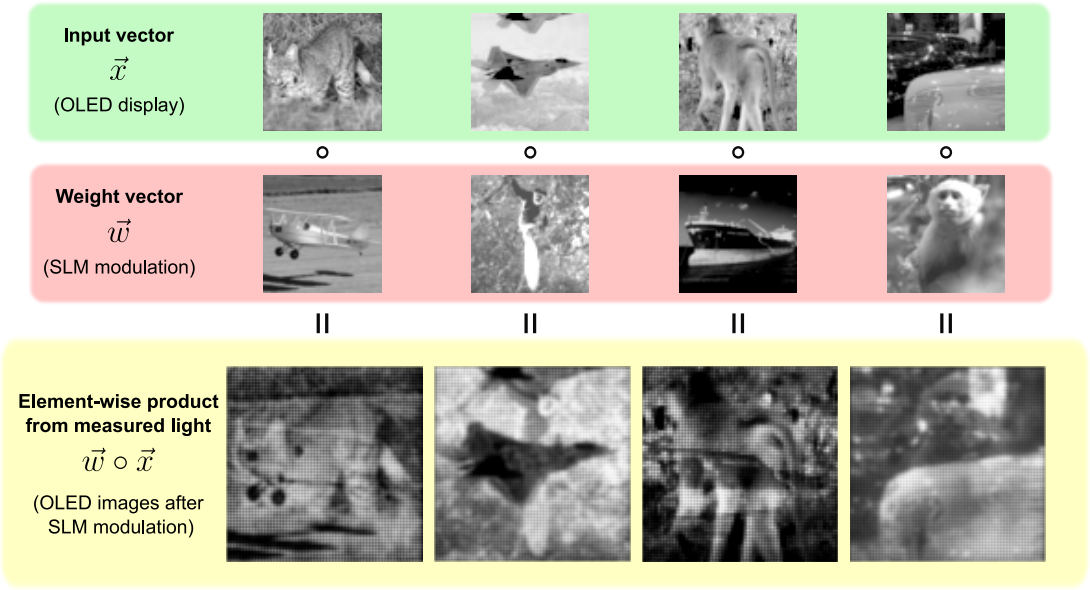


Figure 3.14: **Example measurement of element-wise multiplication between random vectors.** The top two rows show the corresponding input vectors on the OLED display and the weight vectors on the SLM. The bottom row displays modulated light captured by a camera. The input and weight vectors were generated from images in the STL10 dataset of size 64×64 (or $64 \times 64 = 4,096$ elements). Individual pixels are visible in the captured images.

Data Collection Scheme and Photon Budget Control

In order to study how dot product accuracy changes with photon budget, we used a sensitive detector (MPPC) to measure the integrated optical energy. The optical energy consumed for each dot product computation was controlled by tuning the detector integration times (e.g., Figure 3.11 had an integration time of

150 ns). To get enough statistics for noise distribution under low optical power, each detector readout measurement was repeated T times for each vector pair $\vec{w}$ and $\vec{x}$. To get error statistics representative of general vector pairs, we also repeated the measurement for S randomly generated vector pairs of different sparsity from randomly chosen images, as discussed in 3.5.2 and Figure 3.14). We call this set of vector pairs the calibration dataset, and collected a total of $S \times T$ data points. Detector readout $v_{i,j}$ denotes the jth ($j = 1, 2, \ldots, T$) measurement made on the ith ($i = 1, 2, \ldots, S$) vector pair. For each vector pair, the mean value of the detector readouts $\bar{v}_i = 1/T \sum_{j=1}^{T} v_{i,j}$ was calculated for large enough T to eliminate the noise. The detector readouts were quantified either in optical energy, or in number of photons, which is optical energy divided by the photon energy (i.e., ~0.4 aJ at 525 nm). To enable energy efficiency comparisons between different vector sizes, the total optical energy, or number of photons, detected for each dot product was further divided by the number of multiplications in the dot product.

Calibration of Detector Readouts

A calibration model f was made to convert the average detector readout $\bar{v}_i$ to the dot product result $y_{\text{meas},i}$ as $y_{\text{meas},i} = f(\bar{v}_i)$. The calibration involved plotting the ground truth of the dot product $y_{\text{truth},i} = \vec{x}_i \cdot \vec{w}_i$ versus $\bar{v}_i$, followed by fitting the data points to a linear curve using a least-squares criterion. Figure 3.15 shows an example of data points measured on vector pairs of length $N = 505521$. The calibration curve f is plotted in the dashed red line. The range of y_{truth} was normalized to $[0, 1]$ by rescaling $\vec{x}'$ and $\vec{w}'$, based on their definitions in Eq. 3.1, with a multiplicative factor $1/\sqrt{N}$. With the calibration model, we could read

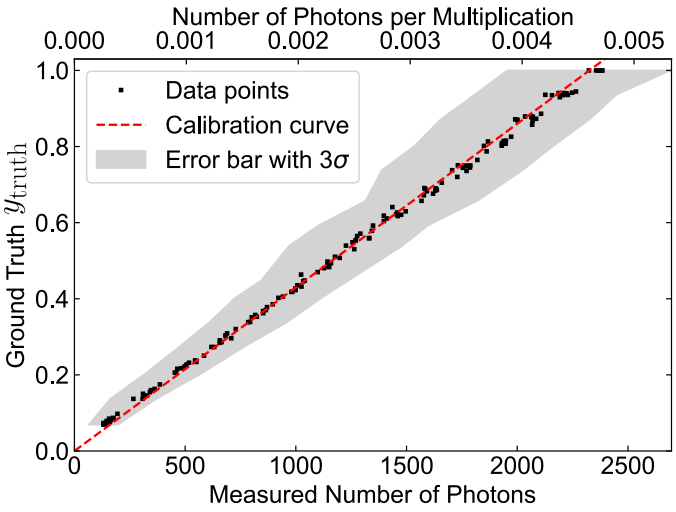


Figure 3.15: **A calibration curve converting detector readouts to dot product results.** The mean detector readout ($\bar{v}_i$, x axis) is plotted against the corresponding ground truth ($y_{\text{truth},i}$, y axis) for every vector pair in the calibration dataset. The unit of detector readout is either the number of photons (bottom axis) or the number of photons per multiplication (top axis). The vector length was 711×711 ($N = 505,521$). The calibration curve (red dashed line) was obtained using a least-squares fit to the data points. The shaded area indicates 3 standard deviations of the repeated measurements. The average number of photons per multiplication of the data points in the plot is 0.0025.

out the dot product result based on the detector readout value. In principle, the calibration only needed to be performed once with the calibration dataset, unless the setup changed (e.g., adding extra attenuation) or has drifted over time.

Quantification of Single-Shot Dot Product Computation Error

After obtaining the calibration curve, we generated another random vector pair test dataset in order to quantify the error statistics of dot product computation performed by our setup. Error was defined as the difference between the mea-

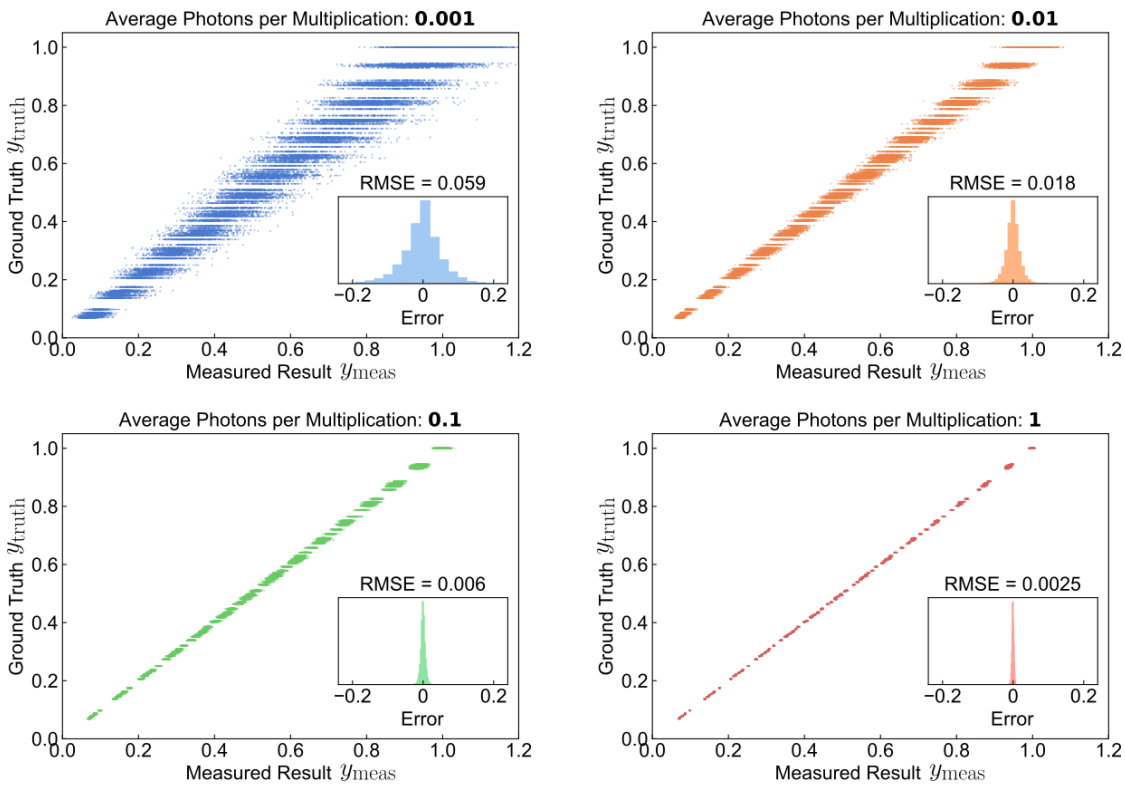


Figure 3.16: **Dot product error analysis at 4 typical photon budgets.** The dot products were computed with vector size $N = 711 \times 711 = 505,521$). The average number of photons per multiplication (indicated at the top of each plot) was controlled by the integration time of the MPPC detector, and averaged over the entire test dataset. For each vector pair, the measurement $y_{\text{meas},i,j}$ was repeated multiple times, and all the data points were plotted. The ground truth $y_{\text{truth},i}$ is plotted against the corresponding measurements $y_{\text{meas},i,j}$. The histogram of errors $y_{\text{truth},i} - y_{\text{meas},i,j}$ is shown in each inset. The overall accuracy representative of the dataset is characterized by the root-mean-square error (RMSE), which is also similar to the value of the standard deviation of the error distribution. The color code is the same as that in Figure 3.13.

sured result and ground truth $y_{\text{truth}} - y_{\text{meas}}$. Suppose we have S vector pairs in the data set and each is repeated T times. For each detector readout $v_{i,j}$ ($i = 1, 2, \ldots, S$, $j = 1, 2, \ldots, T$), $\text{Error}_{i,j} = y_{\text{truth},i} - y_{\text{meas},i,j}$, where $y_{\text{meas},i,j} = f(v_{i,j})$. Unlike calibration, here we used single-shot readouts $v_{i,j}$ rather than the mean

value $\bar{v}_i$.

For each vector pair $\vec{x}_i$ and $\vec{w}_i$, the root-mean-square (RMS) error for different measurement trials was calculated as $\text{RMSE}_i = \sqrt{\frac{1}{T}\sum_{j=1}^{T}\text{Error}_{i,j}^2}$, which can be interpreted roughly as the most likely error one would get from a single-shot computation for vector pair index i. The total RMS error across different vectors in the dataset was calculated as $\text{RMSE} = \sqrt{\frac{1}{S}\sum_{i=1}^{S}\text{RMSE}_i^2} = \sqrt{\frac{1}{ST}\sum_{i,j}\text{Error}_{i,j}^2}$, which could be interpreted as the most likely error one would get from a single-shot computation by randomly selecting a vector pair from the entire test dataset. Histograms of the errors of the test dataset are shown in Figure 3.16 insets.

The scatter plots in Figure 3.16 show how well the computed dot product results matched the ground truth, under different photon budgets. The average number of photons detected during these experiments were different, since they were determined by detector integration time. With higher photon budgets, the error decreases as the noise contribution to the error decreases. For a higher photon budget (>1 photon per multiplication), the RMS error stops decreasing and is instead limited by systematic error due to imperfections in the setup. The four scatter plots in Figure 3.16) correspond to the total RMSE data point of the same color in Figure 3.13b.

3.6 Optical Fan-In and Detection Energy Consumption

We discussed how optical fan-in plays an important role in noise reduction during optical dot-product computation. Noise reduction is possible through the aggregation of a large number of terms in the element-wise vector-vector product. Since the signal-to-noise ratio is almost determined by the total photons re-

quired for a desired output, as vector size increases, the number of photons *per multiplication* decreases. Therefore, for extremely large vector sizes, it is possible to use even less than one photon per multiplication. In this section, we focus on other technical aspects of the optical fan-in operation, including its working principles, additional reasons why it has energy consumption benefits over digital accumulation operations, and generalization to other ONN platforms.

In this experiment, the optical fan-in operation carries out the summation in vector-vector dot product computation by focusing optical spatial modes onto the active area of a detector. More generally, equivalent operations (e.g., weighted optical fan-in, weighted sum of neural activation) exist universally in different spatial-domain optical processing schemes, including the Stanford matrix-vector multiplier [64], volume holography [65], $4f$ convolution [63, 57], and diffractive neural networks [18, 13]. In these schemes, each detector unit reads out the amount of optical energy contained in multiple superimposed spatial modes. The intensity or amplitude of each spatial mode can represent a scalar of some up-stream element-wise product between vectors. Therefore, the energy benefits achieved with optical fan-in are potentially applicable to different spatial domain optical computing schemes, and the energy advantage would improve with the number of spatial modes summed on each single detector provided the noise of each spatial mode is statistically independent of each other.

Compared to digital summation, which sums the readouts from different detector pixels with digital electronic circuits, optical fan-in implements summation through the accumulation of photoelectrons generated in the same piece of detector material. Optical fan-in reduces energy consumption by skipping the

digital summation circuits and reducing the number of pixels, along with their associated amplification circuits. Furthermore, compared to analog electronic summation via pooling of all the photoelectrons generated across a large-area detector, optical fan-in allows all the photoelectrons to be generated in the vicinity of a small-area detector simply by focusing light. In such a way, the energy associated with charging and discharging the detector can be further reduced by shrinking the size as well as the capacitance of the detector. Incidentally, to achieve this effect, the operation of optical fan-in does not require the combination of spatial optical modes.

Optical fan-in not only results in energy savings during the accumulation operation, but also reduces the energy costs of digital memory used for storing intermediate results. In the operation of typical neural networks, memory-associated energy costs often account for the majority of the total energy cost, exceeding even the energy spent on arithmetic operations [11]. Despite attempts to improve energy efficiency in digital electronic processors through the tailoring of data flow to machine-learning tasks [11] or incorporation of more on-chip memory [73], there is still a significant amount of memory that can theoretically be further reduced. For example, in systolic arrays, vector-vector dot products are computed by adding $w_i x_i$ ($i = 1...N$) term-by-term to a partial sum, with each term requiring N dedicated memory units in order to read and write a partial sum only once. Digital electronic adders usually have a small number of input operands, and cannot perform the summation of a large number of terms without saving the intermediate results. In comparison, optical fan-in can physically implement summation of a large number of terms (10^3-10^5) in a single step, which exempts the use of any intermediate memory units, potentially leading to substantial energy savings.

In this experiment, optical fan-in was performed by an objective lens, which projected a de-magnified image of the SLM onto the active area of the MPPC. The de-magnification factor from the SLM to the detector was 0.276×, and the total de-magnification factor from the OLED display to the detector was ~ 0.0442×. In other words, each OLED pixel of original pitch 57.5 μm was imaged to a size of 2.54 μm on the detector. Therefore, the entire 711 × 711 pixel array could be projected onto the detector within a square of size 1.806 mm × 1.806 mm, which fits comfortably within the 3 mm × 3 mm active area of the MPPC. During ideal optical fan-in, all spatial modes associated with a single dot product should be integrated by a single piece of detector material. In this experiment, since the MPPC consists of multiple photodiodes, the spatial modes were not all able to be focused onto a single photodiode (however, it should be noted that the photocurrents of these pixels were still summed as analog signals to form a single output, and no digital operations were involved). Each photodiode covered $(50\,\mu m/2.54\,\mu m)^2 \sim 388$ spatial modes for in-pixel summation. Even with so many spatial modes, no saturation was observed. This is because the photon flux was extremely low for each spatial mode, and thus the simultaneous arrival of multiple photons was rare. The photoelectrons generated by different photodiodes were superimposed as a single analog voltage signal at the detector output.

The energy consumption of optical fan-in can be calculated based on the power consumption of the detector, and can be viewed as the energy cost of analog summation/accumulation. In this experimental setup, the MPPC works at a typical wall-plug power of 1.1 W. Each photodiode consumes, on average 0.3 mW, after dividing the wall-plug power by the total number of the photodiodes (3,600). During a short detector integration time of 100 ns, each photodiode

consumes $0.3\,\text{mW} \times 100\,\text{ns} = 30\,\text{pJ}$ per readout cycle. Dividing this by the number of spatial modes captured by each photodiode yields the energy consumed for each addition. In this experiment, the minimum detector energy consumption of adding one spatial mode to the dot product sum was calculated to be $30\,\text{pJ}/388 = 77\,\text{fJ}$.

In principle, the energy budget for each addition can be further reduced via a number of means: improving the optical fan-in implementation, using smaller focal spots or lower detector gain, and increasing the number of spatial modes for in-pixel summation. Here, we review in detail the effects of each strategy:

1. The focal spot can be reduced to $\lambda/2$ in air, based on the diffraction limit (Abbe's resolution limit with a numerical aperture of 1), which makes each spatial mode occupy $(\lambda/2)^2 = (532\,\text{nm}/2)^2 = 0.071\,\mu\text{m}^2$ area on the detector. The energy consumption for each addition scales with area, thus reducing the focal spot by half would require only $77\,\text{fJ} \times (0.532\,\mu\text{m}/2/2.54\,\mu\text{m})^2 = 0.84\,\text{fJ}$. In addition, the area of each spatial mode can be further reduced by focusing light in materials with refractive index larger than 1 (e.g., glass).
2. In this experiment, the photodiodes were operated with a high gain (Geiger mode) to provide extremely high SNR for single photon detection. In practice, a lower gain could provide two benefits. First, a lower gain can further reduce the energy cost per addition. A lower gain can also prevent saturation of the detector, and allows the accumulation of more terms in a dot product, which is essential for an optical energy advantage. Ideally, the detector volume should be minimized to reduce capacitance, which in turn reduces thermal noise and potentially allows for a high voltage across the detector, i.e. exempting the use of amplifiers [24, 74], Meanwhile, the

number of carriers scales with the detector volume, which results in a limited full-well capacity for a small detector volume. A high gain results in each photoelectron being amplified into many electrons and quickly depletes the carriers in the detector volume. For this reason, the gain of the detector should be reduced until just enough to amplify the signal to overcome thermal noise. For a detector area of only one spatial mode (assuming $\lambda/2$ focal spot size of 525 nm), the RMS value of thermal noise electrons is calculated as $\sqrt{kTC} = \sqrt{4.14 \times 10^{-21}\,\mathrm{J} \times 27.5\,\mathrm{aF}} = 3.38 \times 10^{-19}\,\mathrm{C} = 2.1e^{-}$ at room temperature [24, 74]. At a detection level of 1 photon per spatial mode, a moderately low gain can be applied such that the number of electrons generated by each photon is just large enough in comparison to the number of noise electrons. Such an optimization strategy has also been mentioned in Ref. [47].

3. The SNR of photon detection can also be improved by increasing the number of spatial modes for in-pixel summation. Even though both detector area and capacitance scale with the number of spatial modes to be summed (N), the overall noise scales with $\sqrt{N}$ while the total signal photons scale with N. For example, with 0.5 signal photons (without any detector gain) and 2 noise electrons per spatial mode, the SNR = 0.25 for each spatial mode; with 10,000 spatial modes, the SNR can be enhanced to $0.25 \times \sqrt{10,000} = 25$. Therefore, even in the thermal noise-limited (as opposed to shot noise-limited) detection scheme, it is possible to anticipate less than 1 detected photon for each accumulation in the dot product computation, for sufficiently large vector size.

Through a combination of the optimization measures mentioned above, the energy consumption for optical fan-in can be reduced to the level of 100 aJ

per addition, which is substantially more efficient than digital electronic implementation of accumulation operations (e.g., an 8-bit digital addition with two operands costs $\sim 10\,\text{fJ}$ [75], and reading/writing each intermediate partial sum costs additional 10s fJ, at least).

The energy analysis of the optical fan-in operation can be generalized beyond the accumulation of incoherent spatial modes. When multiple *frequency* or *temporal* modes are impinging onto the same detector, the total optical energy measured by the detector still equals the sum of the optical energy in each individual (frequency or temporal) mode [24, 25, 16, 17, 47], even if coherent light is used — this is due, of course, to energy conservation. More concretely, the energy of frequency and temporal modes can sum up in the same fashion as incoherent spatial modes, because light pulses of different wavelengths or pulses arriving at different times at the same detector do not interfere with each other. In principle, the energy cost associated with summing temporal or frequency modes should be no more than summing spatial modes, since the detector size (capacitance, and thermal noise) no longer needs to scale with the number of modes, as discussed before in this section. More detailed discussion on the energy scaling on these different ONN platforms can be found in, for example, Ref. [47, 74, 24].

3.7 Comparison to the Stanford Matrix-Vector Multiplier

Our setup can be viewed as a generalization of the classical Stanford matrix-vector multiplier, whose operation can be summarized in three steps: fan-out, element-wise multiplication, and fan-in (Figure 3.17). The major difference be-

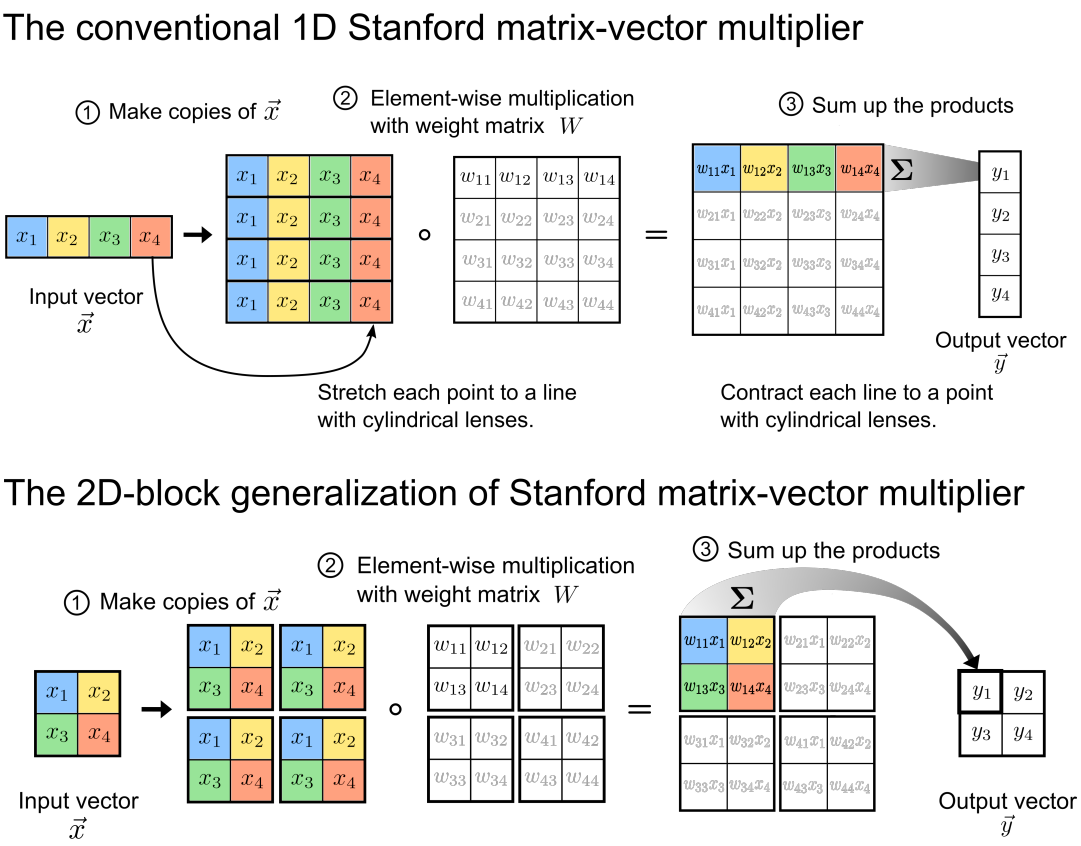


Figure 3.17: **The comparison between the classical Stanford matrix-vector multiplier and our 2D-block scheme.** The top row shows how the classical Stanford Matrix-vector Multiplier performs matrix-vector multiplication in a sequence of three steps of optical fan-out, element-wise multiplication, and optical fan-in. The bottom row shows how the 2D-block scheme performs the same corresponding operations.

tween our setup and the Stanford matrix-vector multiplier lies in the geometric arrangement of the input vector and its corresponding weights (the weight vector) in 2D blocks instead of 1D arrays, which lead to different physical implementations. Compared to 1D arrays, the 2D-block arrangement of both input and weight vectors offer advantages in scalability, especially for vector-vector dot products with a large vector size.

The 2D-block arrangement is especially suitable for applications in image

classification, where the input data are usually 2D images of high dimensions. For natural-scene images, and the weights trained for them by neural networks, (e.g., see Figure 4.3), the pixel values usually do not vary drastically in local areas except for few high-contrast boundaries. Preserving smooth local features in the 2D form reduces errors resulting from minor shifting or blurring of the input and weight vectors. Therefore, our setup is more tolerant of minor errors in instrumentation, such as imperfect imaging or crosstalk between SLM pixels (Figure 3.18a). In comparison, flattening the vectors and weights into 1D arrays, as required by the Stanford matrix-vector multiplier, would break the smoothness of local features, which can result in more severe errors (e.g., the crosstalk level between the neighboring SLM pixels increases with the level of contrast between them).

The 2D-block arrangement also reduces the amount of crosstalk between different input or weight vectors tiled next to each other (Figure 3.18b). For each block, crosstalk occurs to the pixels in contact with adjacent blocks. When a vector of size N is wrapped into a square, the total number of pixels on its perimeter equals $4\sqrt{N}$ (Figure 3.18b top panel). If the vector were instead shaped into a 1D array, it would share a boundary of a total of $2N$ pixels with neighboring arrays (Figure 3.18b bottom panel). For a large vector of size N, 2D blocks can potentially reduce the amount of crosstalk by over an order of magnitude more than 1D arrays, due to the difference in scaling of $\sqrt{N}$ (Figure 3.18b). Even though a gap can be introduced between the regions corresponding to different vectors, it would also reduce the fill factor as well as the computational throughput. Therefore, through decreasing the surface-to-volume ratio, the 2D-block arrangement can effectively reduce crosstalk between different vectors with a high fill factor on both OLED and SLM panels.

It should be noted that in the classical Stanford matrix-vector multiplier both optical fan-out and fan-in are implemented with cylindrical lenses as reverse processes of each other. In the 2D-block arrangement, this symmetry is broken because the optical fan-out process needs to create copies of the same block, while the optical fan-in focuses all elements in each block. In our demonstration, we performed the fan-out operation digitally by displaying different copies of the same vector on the OLED display, which was sufficient for the purpose of demonstrating sub-photon scalar multiplication. For the 2D-block scheme, it is possible to implement optical fan-out with several techniques, including imaging with a microlens array [76, 55] or beam splitting using an array of beam splitters [77]. The optical fan-out operation would allow a very low number of photons in each spatial mode: the optical fan-out splits the photons emitted from a reasonably weak light source and distribute each portion of the photons into many spatial modes, with each mode performs a scalar multiplication. In addition, since each spatial mode is the result of many vacuum modes being coupled to the original spatial mode populated by a single light source, the noise of each spatial mode can be reduced to be quite close to shot noise, even if the light source intensity is not perfectly stable.

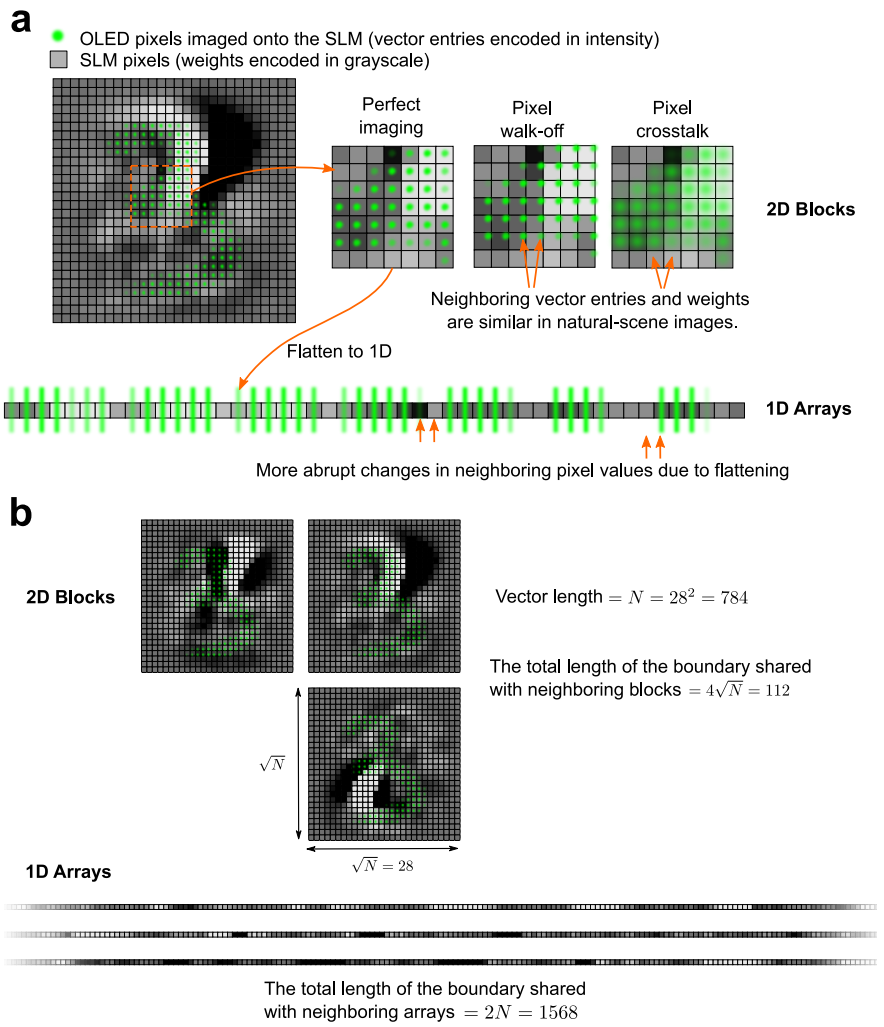


Figure 3.18: **Comparison between the 2D-block and 1D-array representation of the input vector and weight matrices.** **a,** Error reduction effects of 2D-block representation of natural-scene images. The smooth local features intrinsic to the image data and corresponding weight matrices mitigate the crosstalk between neighboring pixels caused by imperfections in imaging (e.g., degradation of focal qualities) or instrumentation (e.g., crosstalk between SLM pixels). Reshaping 2D images into 1D arrays breaks the continuity intrinsic to these images. **b,** The 2D-block representation reduces the crosstalk between neighboring vector and weight blocks. The 2D blocks share substantially shorter boundaries with each other ($\propto \sqrt{N}$) compared to 1D arrays ($\propto N$), which helps to reduce crosstalk between different vector and weight blocks, or to reduce the gaps between them for a higher fill factor on the SLM.

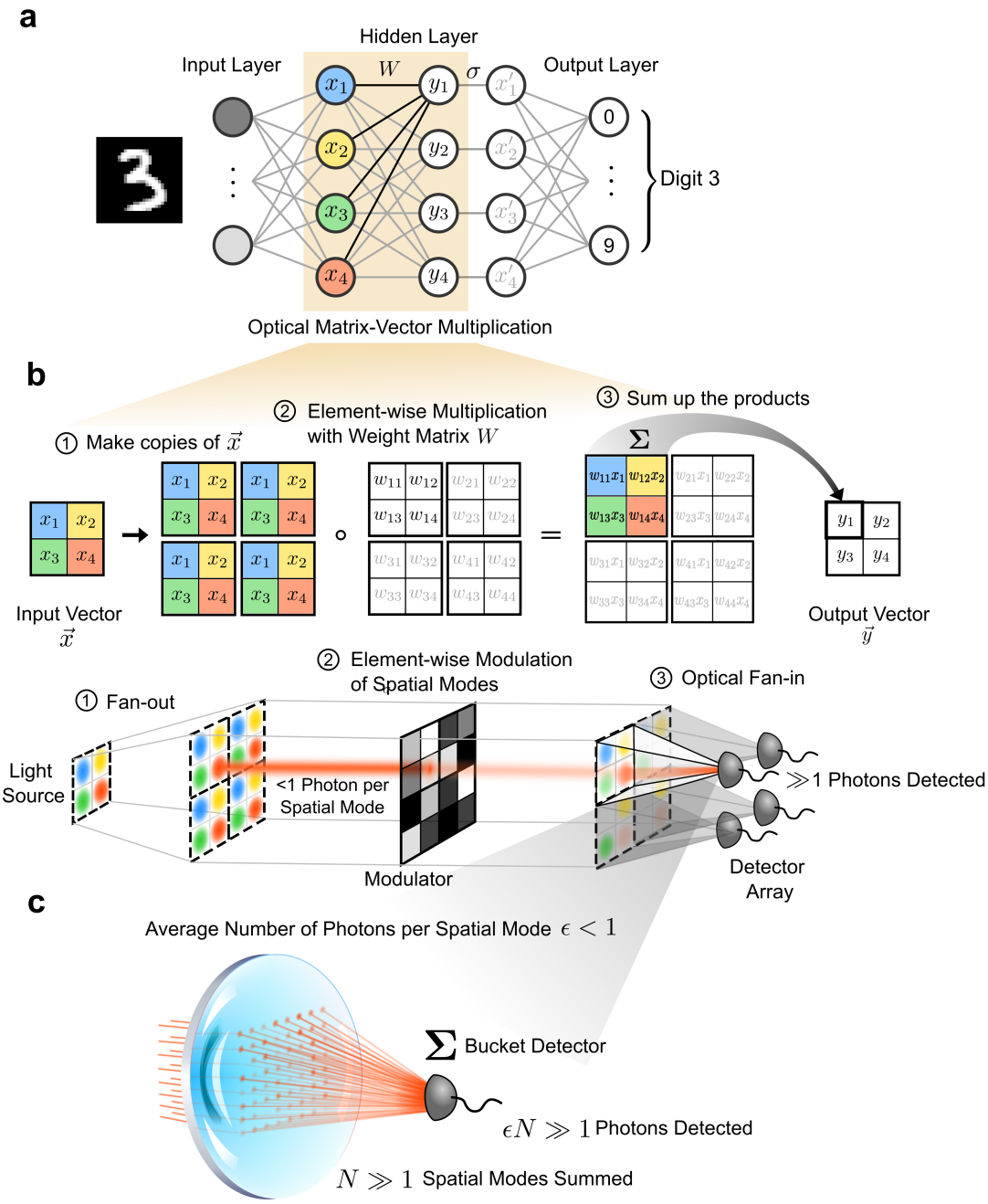
CHAPTER 4

NOISE-RESILIENT OPTICAL NEURAL NETWORKS USING < 1 PHOTON PER MULTIPLICATION

Much of the content in this chapter is adapted from the work presented in Wang et al. [1]. My major contributions to this work included designing and performing the experiments, analyzing the data, and preparing the paper alongside the other co-authors.

4.1 Background

Theoretical studies [25, 24] show that the energy consumption of optical matrix-vector multiplication can be orders of magnitude lower than what is possible in electronics. It has been predicted that, for sufficiently large vector sizes, the optical energy cost of each scalar multiplication can be amortized in matrix-vector multiplication to achieve a level as low as less than 1 photon per scalar multiplication [24]. With this optical matrix-vector multiplier we reported in the last chapter, we can construct an optical neural network (ONN) that performs image classification using less than 1 photon per scalar multiplication, matching theoretical predictions. However, even if the theory work has promised this possibility, there are technical challenges to tackle down the presence of noise and errors in this situation. We will introduce how to train and implement these noise-resilient ONNs in the later sections of this chapter.



4.2 Large-scale Optical Matrix-Vector Multiplication

The optical processor we designed and constructed uses the following scheme to perform matrix-vector multiplications $y = W\vec{x}$. Each element x_j of the input

Figure 4.1: **A 3D free-space optical matrix-vector multiplier that uses less than one photon per scalar multiplication performed.** **a,** The role of optical matrix-vector multiplication in executing the forward-pass (inference-mode) operation in a fully connected neural network. The matrix multiplications involved in propagating each layer forward are performed optically and the element-wise nonlinear activation functions are performed electronically. Each neuron in the middle (hidden) layer is color-coded to show the correspondence between the mathematically abstract neurons in (a) and their optical implementation in (b). **b,** A step-by-step illustration of how optical matrix-vector-multiplication can be performed by decomposing the matrix-vector operation $W\vec{x}$ into blocks of vector-vector dot products that are implemented as scalar multiplications performed in parallel, followed by summations (accumulations) performed in parallel. The depiction is for a 4×4 matrix and a 4-dimensional vector, but the concept extends naturally to matrices of arbitrary dimension. The top row shows mathematically abstract operations, and the bottom row shows the corresponding physical operations with optics. "$\circ$" denotes element-wise multiplication between two matrices or vectors of the same size. Individual scalar multiplications are realized optically by encoding one operand (x_j) in the intensity of a single spatial mode (depicted as a beam for illustrative purposes), and the other operand (w_{ij}) in the transmissivity of a single pixel of a modulator: propagation of the beam through the modulator's pixel yields the scalar multiplication, whose result $w_{ij}x_j$ is encoded in the intensity of the transmitted light. Each summation $\sum_j w_{ij}x_j$ is performed optically by focusing multiple spatial modes onto a single detector. **c,** An illustration of how optical matrix-vector multiplication can consume less than 1 photon per scalar multiplication when for a large vector size. A single lens is used to sum the intensities of the spatial modes encoding the element-wise products $w_{ij}x_j$ between the ith row of W and the vector $\vec{x}$. For a vector of size N, there are N spatial modes whose intensities are summed. If N is sufficiently large then even if each individual spatial mode contains $\epsilon < 1$ photon on average, the total number of photons impinging on the detector will be $\epsilon \gg 1$, allowing high signal-to-noise-ratio measurement of the summation result $y_i = \sum_j w_{ij}x_j$.

vector $\vec{x}$ is encoded in the intensity of a separate spatial mode illuminated by a pixel of a light source, and each matrix element w_{ij} is encoded as the transmissivity of a modulator pixel. We used an organic light-emitting diode (OLED) display as the light source and a spatial light modulator (SLM) for intensity modulation. Matrix-vector multiplications were computed in three physical steps: 1) Fan-out: The input vector's elements were spatially arranged into a 2D block (Figure 4.1b, top left). The 2D block representing $\vec{x}$ was replicated a number of times equal to the number of rows in the matrix W, and then tiled on the OLED display as shown in Figure 4.1b (top row). 2) Element-wise Multiplication: Each OLED pixel encoding a single vector element x_j was aligned and imaged to a corresponding pixel on the SLM, whose transmissivity was set to be $\propto w_{ij}$. This performs the scalar multiplication $w_{ij}x_j$ (Figure 4.1b bottom middle). 3) Optical Fan-in: The intensity-modulated pixels from each block were physically summed by focusing the light transmitted by them onto a detector. The total number of photons impinging on the ith detector is proportional to the element y_i of the matrix-vector product $y = W\vec{x}$ (Figure 4.1b bottom right). We can interpret each y_i as the dot product between the input vector $\vec{x}$ and the ith row of the matrix W.

This design computes all the scalar multiplications and additions involved in a matrix-vector multiplication in parallel in a single pass of light through the setup. The encoding of vector elements in optical *intensity* constrains the setup to performing matrix-vector multiplications with matrices and vectors that have *non-negative* elements. However, we can use the system to perform matrix-vector multiplications with matrices and vectors that have *signed* elements (elements that can take both positive and negative values) by using offsets and scaling to convert the calculations to matrix-vector multiplications in-

volving only non-negative numbers [64].

Our 2D-block design can be, and was, implemented with spherical-lens systems, which are well-corrected for errors in large-field-of-view imaging. The setup enabled us to align an array of 711×711 pixels on the OLED display to an array of 711×711 pixels on the SLM (Sections 3.4.1, 3.4.2, and 3.4.3), allowing 711×711=505,521 scalar multiplications to be performed in parallel. Our experimental setup was, with a single pass of light through the setup, capable of performing matrix-vector multiplications with matrices having arbitrary dimensions, subject to the total number of matrix elements being no larger than 505,521. In the special case of the matrix merely being a vector, our setup performed a single vector-vector dot product with vectors sizes up to 505,521.

The ability to perform dot products between very large vectors with our block-encoded design enabled us to achieve extremely low optical energy consumption. For each vector-vector dot product that the system computes, the summation of the element-wise products is performed by focusing the spatial modes corresponding to the element-wise products onto a single detector. If the vectors have size N, then N spatial modes are incoherently summed on the detector.

Consequently the detector's output, which is proportional to the dot-product answer, has a signal-to-noise ratio (SNR) that scales as $\sqrt{N}$ under the shot-noise limit [25]. If the vector size is sufficiently large then even if the individual spatial modes each have an average photon number far less than 1, the total number of photons impinging on the detector can be much greater than 1, and so precise readout of the dot-product answer is possible (Figure 4.1c).

4.3 ONN Using Sub-Photon Multiplications

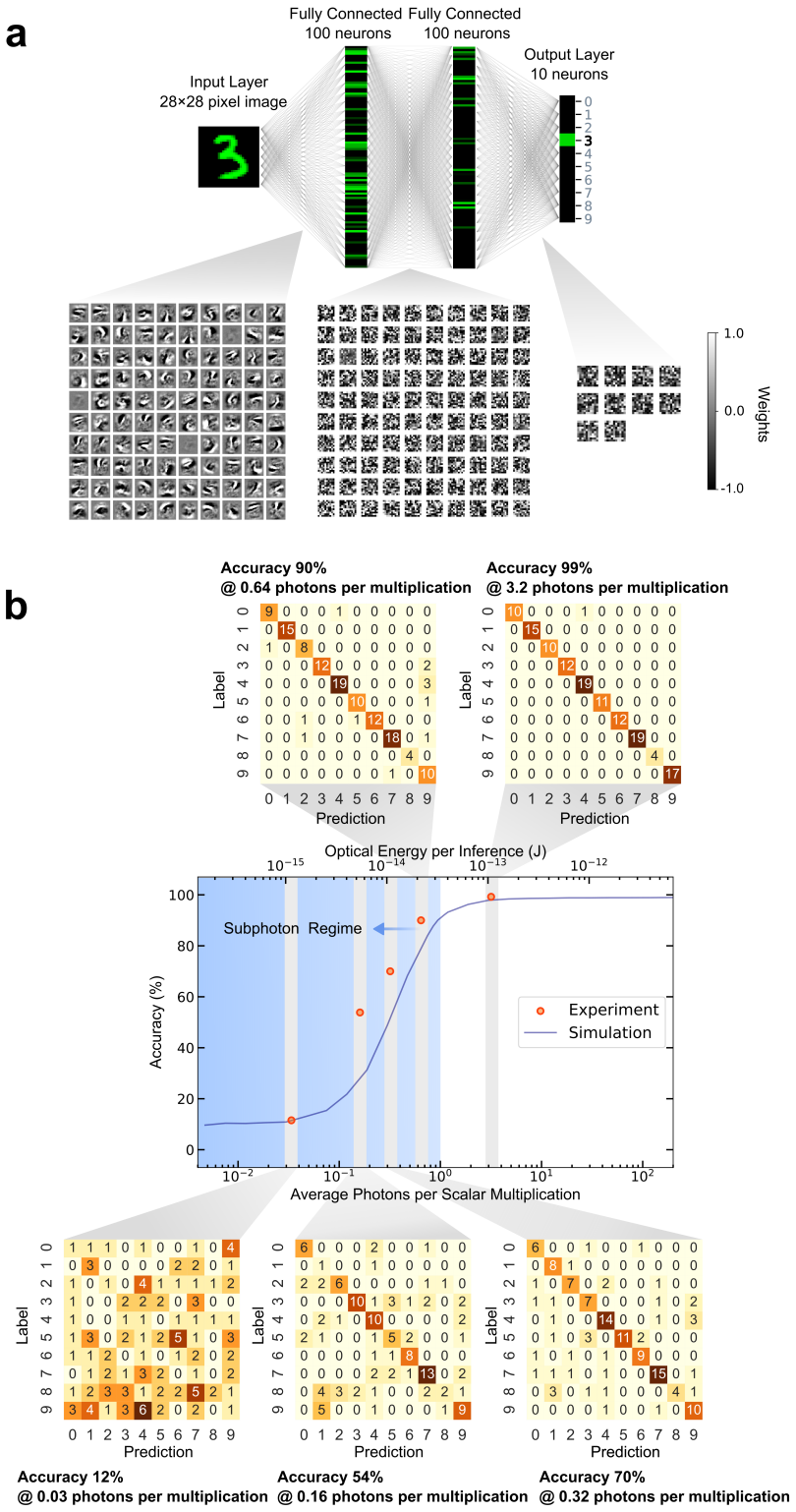


Figure 4.2: **MNIST handwritten-digit classification demonstrated with an optical neural network using less than one photon per scalar multiplication. a,** Illustration of the neural-network architecture for handwritten-digit classification that we implemented as an ONN. The neural network is composed of a sequence of fully connected layers: an input layer consisting of 28×28 neurons encodes the image to be classified (left); two hidden layers consisting of 100 neurons each (middle), and an output layer containing 10 neurons (right). The activation of each neuron is illustrated as green brightness for the example of an input image containing the digit 3 (vertical bars, top panel). The weights of the connections between neurons for all four layers are shown (grayscale square arrays, bottom panel); each square array shows the weights *from* all the neurons in one layer *to* one of the neurons in the next layer. **b,** Classification accuracy tested using the MNIST dataset as a function of optical energy consumption (middle panel), and confusion matrices of each corresponding experiment data point (top and bottom panels). The *optical energy per inference* is defined as the total optical energy received by the photodetectors during execution of the three matrix-vector multiplications comprising a single forward pass through the entire neural network.

Having characterized the accuracy of our experimental setup for performing multiplication operations with random vectors, we set out to demonstrate its use as the core of an experimental ONN implementation. We realized an ONN comprised of fully connected layers where the matrix-vector multiplications between each layer were performed optically using our experimental setup, and where the nonlinearity was applied electronically (using a digital processor) between each layer.

Our main goal was to determine the extent to which our ONN could tolerate multiplication inaccuracy resulting from the use of a very limited photon budget. Our approach was to run a trained neural network with our setup and measure the classification accuracy as a function of the number of photons

used. We used handwritten-digit classification with the MNIST dataset as our benchmark task and trained a four-layer fully connected multi-layer perceptron (MLP) (Figure 4.2a) with a back-propagation procedure designed for use with low-precision inference hardware (quantization-aware training—QAT [78]; see Section 4.4).

We evaluated the first 130 test images in the MNIST dataset under 5 different photon budgets: 0.03, 0.16, 0.32, 0.64, and 3.2 photons per scalar multiplication (Figure 4.2b, center panel, orange dots). We found that using 3.2 photons per multiplication led to a classification accuracy of ~99% (Figure 4.2b, top-right panel), which was almost identical to the accuracy (99%) of the same trained neural network run on a digital computer. In the sub-photon regime, using 0.64 photons per multiplication, the ONN achieved >90% classification accuracy (Figure 4.2b, top-middle panel). The experimental results agree well with the results from simulations of the same neural network being executed by an ONN that is subject to shot noise (Figure 4.2b, center panel, dark-blue line). To achieve an accuracy of 99%, the total optical energy *detected* for each inference of a handwritten digit was ~1 pJ (Figure 4.2b). For the weight matrices used in these experiments, the average SLM transmission was ~46%, so when considering the unavoidable loss at the SLM, the total optical energy needed for each inference was ~2.2 pJ. For comparison, 1 pJ is close to the energy used for only a single scalar multiplication in electronic processors [73], and our model required 89,400 scalar multiplications per inference (see Section 4.5).

4.4 Training Protocol of Noise Resilient Optical Neural Networks

For handwritten digit classification (MNIST database), we trained a 4-layer neural network with full connections (i.e., a multilayer perceptron, MLP). The inputs are 8-bit grayscale images of size $28 \times 28 = 784$ total pixels, followed by two fully-connected hidden layers, each comprising 100 neurons with ReLU as the nonlinear activation function. The output layer has 10 neurons, with each neuron corresponding to a digit from 0 to 9. The neural network was implemented and trained in PyTorch (1.7.0). To improve the robustness of the model to numerical error, we employed several techniques during training:

1. Quantization-aware training (QAT): The activation of neurons were quantized to 4 bits and weights to 5 bits to adapt to the numerical precision of the setup. For example, for vector size of 784, we found the SNR of dot products is equivalent to ~4-bit numerical precision. Even though the SLM could be controlled with 8-bit numbers, we decided to quantize its weight to 5 bits, which matched better with its extinction ratio of 50 and did not seem to have any negative impact on MLP accuracy. To reduce the numerical sensitivity caused by quantization (e.g., in the regular nearest neighbor scheme, 0.49 is rounded down to 0, while 0.51 is rounded up to 1), we used random digitization between the adjacent levels, which was observed to improve the model robustness against random noise. We found that a few (6-12) warm-up training epochs with full 32-bit float precision helped to protect the model from aggressive quantization in the initial stage, after which the application of quantization noise was less likely

to derail the training process, but still helped to fine-tune the parameters.

2. Data augmentation with random image transform and convolution: To improve model tolerance to potential hardware imperfections, we imitated similar kinds of errors on input images. For example, the misalignment was modeled as random rotation (within ±5°) and translation (±4% of image size in any direction), mismatched zoom factor as random zooming (±4% of image size), and intra-pixel crosstalk as a mild convolution with a 3×3 blurring kernel. We observed that these measures not only helped to improve model immunity to imaging error, but also improved overall model accuracy and robustness to photon noise by reducing overfitting with regularization. The data augmentation was only performed on the input layer rather than all hidden layers, due to computational complexity and the observation that hidden layers were usually more sparse, making crosstalk between neighboring pixels was less of an issue.
3. Optimizing training parameters: We used a stochastic gradient optimizer for training with a learning rate typically between 0.03 and 0.05, and a momentum between 0.7 and 1. Learning rate decay was applied every 20 epochs with a decaying rate between 0.3 and 0.5. The training parameters were randomly generated within the range for different trials of training, and fine-tuned by using the package Optuna [79].

The training of each model took 100 epochs. Several hundred models were trained, each with slightly different randomly generated training parameters. Afterwards, each model was executed with simulated shot noise at different photon budgets (code available at: `https://github.com/mcmahon-lab/ONN-QAT-SQL`), and then the model yielding the best accuracy at low photon budgets was chosen to be run on the ONN setup.

It is important to note that the quantization of neuron activations was only performed during training on a digital computer, but not during the inference stage on the ONN. We observed that even though the models were trained with noise, they still performed better when run with full precision. Therefore, the trained weights and neural activations were loaded with the maximum allowable precision for the hardware (i.e., 7 bits for the OLED display, and 8 bits for the SLM). The training code can be found in this GitHub repository: `https://github.com/mcmahon-lab/ONN-QAT-SQL`.

The neural network model was completely trained on a digital computer, without any customization to the optical setup. There was no *in situ* re-training of any model parameter, nor any addition of extra digital layers to assist with the optical setup. To be clear, such digital assistance is extremely useful for bridging the gap between hardware and software in real-world applications. However, they were not used for this study because they would have affected the one-to-one correspondence between optical and digital operations, and therefore our ability to unambiguously quantify the number of optical operations — and the amount of optical energy used — for each optical operation.

4.5 Workflow for Running Optical Neural Networks for Inference

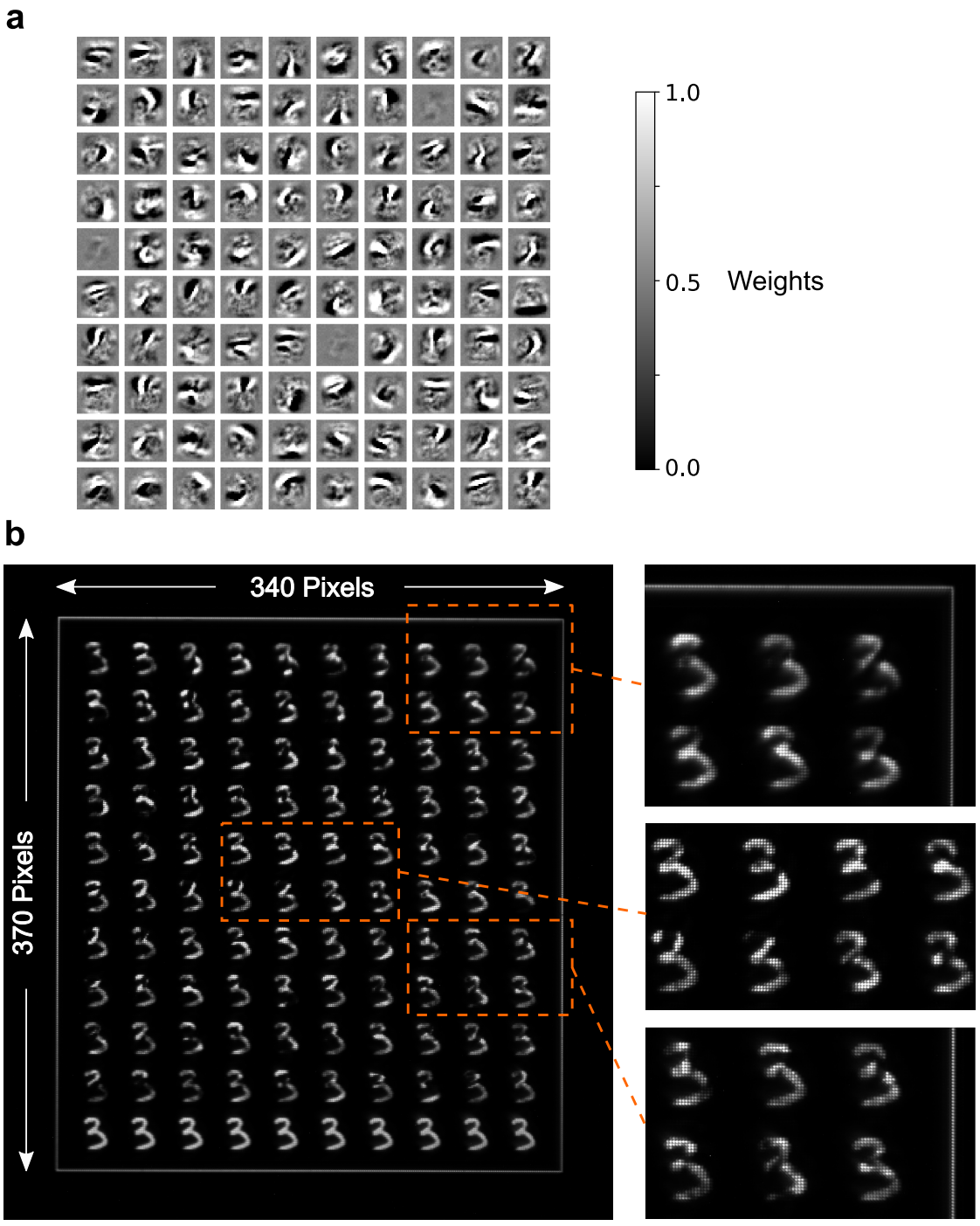
The trained neural network (MLP 784-100-100-10 as described in 4.4) was executed for inference with our optical vector-vector dot product multiplier in the following steps:

1. Starting from the input layer, the matrix-vector multiplication involved in the forward propagation from the current layer to the next layer was computed optically by each dot product, according to the procedure described in 3.5.1. The matrix weights loaded onto the SLM were exactly the same as those in the neural network trained on the digital computer. The number of photons per multiplication in matrix-vector multiplication was controlled by adjusting the number of detector samples to sum.
2. The bias terms and the nonlinear activation function were applied digitally to the matrix-vector multiplication result, and these parameters followed exactly as those in the trained neural network, without any modification or retraining. The resulting neuron activations were used as the input vector to the matrix-vector multiplication that leads to the next layer (go back to step 1) unless there is none (go to step 3).
3. At the output layer, the prediction was made based on the highest score.

For step 1, since the inputs and neural activations were both non-negative due to our choice of ReLU nonlinearity, only the weight matrices needed to be shifted and normalized. The element-wise multiplication of the first layer is visualized in Figure 4.3b, with the weight matrix displayed on the SLM visualized in Figure 4.3a for comparison. Each matrix-vector multiplication $y_i = \sum_j W_{ij} x_j$ was decomposed into vector-vector dot products $\vec{w_i} \cdot \vec{x}$, where $\vec{w_i} = W_{ij}$ $(j = 1, 2, ...)$ encodes the (synaptic) weights from neuron i in the previous layer to neuron j in the next layer, and $\vec{x} = x_j$ $(j = 1, 2, ...)$ encodes the neural activations of the previous layer. Each modulated handwritten-digit image shown in Figure 4.3b illustrates the element-wise product $\vec{w_i} \circ \vec{x}$. To obtain the answer to the matrix-vector multiplication, element-wise modulated spatial modes in each block (i.e., $\vec{w_i} \circ \vec{x}$) were summed up by optical fan-in as described

in 3.6, which is equivalent to summing all the pixels in each block shown in the image taken by the camera in Figure 4.3. The degree of optical fan-in equals the vector size or the number of neurons in the previous layer (i.e., 784 for the first layer, 100 for the second layer etc.). The $\vec{I} \cdot \vec{x} = \sum_j x_j$ term in Eq. 3.2 was computed by adding an additional input vector block and setting the corresponding SLM pixels' transmissivity to the maximum in order to encode $\vec{I}$ (e.g., Figure 4.3b, last row in the image. An entire row was used for illustration purposes and redundancy, while in fact only one additional block was needed for the entire layer.). The integrated optical energy was translated to the answer of the dot product based on a calibration curve, which was made by fitting the measured optical energy to the ground truth of the dot products using the first 10 samples of the MNIST test dataset in a fashion similar to that described in Section 3.5.2.

For each forward propagation of the neural network, $784 \times 100 + 100 \times 100 + 100 \times 10 = 89,400$ multiplications and 89,400 additions were performed optically. The total digital assistance for each forward propagation include $100 + 100 + 10 = 210$ additions for applying the $\vec{I} \cdot \vec{x}$ term (to shift the dot product between vectors of non-negative elements in order to obtain the dot product between vectors of signed elements), $100 + 100 + 10 = 210$ additions for applying digital biases, and $100 + 100 = 200$ applications of ReLU nonlinear activation functions (involving only simple operations, such as comparing each element to 0 and setting an element to 0 if it is smaller than 0).



4.6 Energy Efficiency of the Optical Neural Network

The total number of optical operations is calculated as the number of operations involved in each matrix-vector multiplication from one neural network layer to

Figure 4.3: **An example of the matrix-vector multiplication results of the first fully-connected Layer of the ONN. a,** Visualization of the weights of the connection between the first and the second layer of neurons; the pixel values in each square array indicate the weights from all the neurons in the first layer to one of the neurons in the second layer of neurons. The weights were implemented as light transmission on the SLM (1 for full light transmission and 0 for no light transmission.). **b,** The images show the results of element-wise multiplication between an input vector ($28 \times 28 = 784$ elements) and each row of the matrix of the first fully-connected layer (100×784) of our ONN model, as captured by a camera. The last row computed $\sum_j x_j$, which was used to offset the output vector for negative elements. Only the output of the first layer is shown.

the next. For an input layer of width N (also referred to as N neurons in this layer) and an output layer width N', the total number of scalar multiplication equals NN' and the total number of scalar addition is $(N-1)N'$. According to 4.5, the four layers of the ONN have width 784, 100, 100, and 10, respectively. For each forward propagation through the neural network, the total number of optical multiplications — defined as scalar multiplication between a pair of non-negative real numbers — is $784\times(100+1)+100\times(100+1)+100\times(10+1) =$ **90,384** (the additional +1 term in each output layer width is due to the calculation of one additional dot product $\vec{1}\cdot\vec{x}$ as an offset term, see 4.5). The total number of optical additions, defined as the addition of a pair of non-negative numbers, is $(784-1)\times(100+1)+(100-1)\times(100+1)+(100-1)\times(10+1) =$ **90,171**. The total digital electronic operations for each forward propagation includes $100+100+10 = 210$ 16-bit additions for applying the $\vec{1}\cdot\vec{x}$ term (to shift the dot product between vectors of non-negative elements in order to obtain the dot product between vectors of signed elements), $100 + 100 + 10 = 210$ 16-bit additions for applying digital biases, and $100 + 100 = 200$ applications of ReLU nonlinear activation

functions (involving, essentially, a comparison between each element and 0, and then setting the element to 0 if it is less than 0). The total count of all operations, including optical and electronic, for a single forward pass of the neural network, is summarized in Table 4.1.

Optical operations account for $100\% - 621/180,555 = 99.66\%$ of the total amount of operations involved in a single forward pass of the neural network to classify a full-resolution 28×28 handwritten-digit image (also referred to as per inference). More importantly, we did not use any additional trainable digital layer to read out results from or adapt to the physical setup. This practice is necessary for the accurate quantification of optical-operation energy efficiency. It is a well-known fact in machine learning that a single linear layer can already achieve 91.6% classification accuracy on the benchmark dataset of MNIST (`http://yann.lecun.com/exdb/mnist/`), and a straightforward implementation of reservoir computer or extreme learning machine with a *random* neural network and a single digital trainable output layer can achieve even 98% accuracy [80]! In the case of optical reservoir computing, the optical layer performs a huge amount of computation with high efficiency, but the computation is almost random and has limited programmability. As a result, a digital output layer is required to extract useful results from optical computation that are relevant to the task at hand. In order to isolate the contributions of digital electronic operations from those of optical operations, we restrained ourselves from using any trainable digital operations to adapt to the experimental setup in this work. The neural network model run on the ONN setup was isomorphic [23] to the one run on a digital computer: each optical operation replaced a corresponding digital operation in the trained neural network model that could otherwise be executed on a digital computer, without changing any parame-

ter (e.g., matrix weights). The absence of any extra trainable digital parameter forced the ONN to solely rely on its optical operations: each optical operation needed to be engineered to be highly effective and faithful to the neural network model in order to contribute to the correct classification result.

The optical energy per operation was quantified according to the following scheme: after optical fan-in, all the spatial modes involved in a vector-vector dot product were focused on a bucket detector (MPPC). The detected optical energy per dot product was measured by integrating optical power over a period of time. The number of photons detected per multiplication was then calculated as:

$$\text{\# of photons detected per multiplication} = \frac{\text{Detected optical energy per dot product}}{\text{\# of multiplications per dot product} \times \text{Single photon energy at } 525\,\text{nm}}$$

The total detected optical energy for each layer during the execution of the ONN for one inference was calculated with the following equation:

$$\text{Total detected optical energy per layer} = \text{Detected optical energy per dot product} \times \text{\# of dot products per layer}$$

The integration time of the detector was typically chosen in the range of 1 µs to 60 µs, in order to keep the number of photons per multiplication approximately constant across different layers. A few different values of the number of photons per multiplication were chosen in order to observe how the classification accuracy of the ONN changes with photon budget. The top section of Table

4.2 shows the average detected optical energy per dot product for each layer of the ONN. The middle section of Table 4.2 shows the detected photons per multiplication by each layer of the ONN, including a weighted average across the entire ONN. The weighted averages were derived by weighting with the total number of multiplications in each layer (listed in Table 4.1). The bottom section of Table 4.2 shows the total detected optical energy for each layer and the entire ONN during one inference, which is also plotted against the top horizontal axis in Figure 4.2b.

When studying the energy efficiency of ONNs, attention should also be paid to the total number of optical operations required to achieve certain level of performance on the benchmark task, in addition to the energy efficiency of each operation [81]. This is because the number of operations of different neural network models to solve the same benchmark problem is generally different, but ultimately the energy efficiency should be determined by the *total* amount of energy to solve the problem instead of the energy consumption of each operation. Take MNIST as an example again, $\sim 10^6$ digital operations are regularly needed to reach 98 $\sim$ 99% classification accuracy with multi-layer perceptrons (`http://yann.lecun.com/exdb/mnist/`). The total number of optical plus electronic operations shown in this work (Table 4.2) falls well within this range. This indicates that the high optical energy efficiency per operation shown in this work readily translates to a high optical energy efficiency per inference (see Table 4.2 bottom section), after multiplying with the total number of operations in the neural network model. The 100-fJ level detected optical energy per inference is several orders of magnitude smaller than the total energy consumption per inference of digital electronic processors that are highly optimized for energy efficiency on the same benchmark tasks (10^3 times smaller than Ref. [82]

with binary weights and activations, $\sim 10^5$ smaller than Ref. [83, 84]). Admittedly, once the optical loss and non-optical energy is included in the total energy consumption of ONNs, the gap between the energy consumption of optical and digital electronic processing would decrease. Nevertheless, the low detected optical energy per inference shown in this work indicates optical processors have the potentials to achieve quite large advantage over electronic processors in terms of the whole-system energy efficiency provided that the entire system including optical and non-optical parts are sufficiently optimized.

The power consumption of each experimental instrument is listed in Table 4.3 for the completeness of experimental details. Note that the *whole-system* energy efficiency derived from these power consumption does *not* carry any significant implication on the overall energy efficiency of the ONN, since none of the devices was customized for optimized power consumption or refresh rate. For one example, based on the amount of digital operation needed, the laptop can be probably replaced by a digital processor that consumes orders of magnitude lower power. For another example, the maximum data throughput rate in this experiment was limited by the refresh rate of OLED at 60 Hz, which was made to adapt to human eye, most likely the speed can be increased several orders of magnitude as well if the device were optimized for optical computing purposes. Due to the large non-optical energy consumption overhead in the hardware that is irrelevant to the ONN computation, the total power consumption of the setup was constant at ~65 W (Table 4.3), regardless of the speed of computation. The maximum achievable computational speed is (711^2 multiplications/frame+ ($711^2 - 1$) additions/frame) $\times$ (60 frames/second) $\sim 6.07 \times 10^7$ operations/s. At this computing rate, the energy efficiency is $(6.07 \times 10^7$ operations/s)/65 W $= 0.93 \times 10^6$ operations/J. Since the experimental setup was built for the purpose

	Layer1 → 2 (784-100)	Layer2 → 3 (100-100)	Layer3 → 4 (100-10)	Whole network
Number of input neurons (dot-product vector size)	784	100	100	—
Number of output neurons (total number of dot products)	100+1	100+1	10+1	—
Total optical multiplications	79,184	10,100	1,100	90,384
Total optical additions	79,083	9,999	1,089	90,171
Digital addition of the offset term $\vec{1} \cdot \vec{x}$ (FP16)	100	100	10	210
Digital addition of model biases (FP16)	100	100	10	210
Digital ReLU nonlinear activation functions	100	100	0	200
Digital max/softmax for a 10-D vector	0	0	1	1
Total number of optical operations	158,267	20,099	2,189	180,555
Total number of digital operations	300	300	21	621

Table 4.1: The number of all operations during the execution of the ONN for one inference

of studying the optical energy efficiency of ONNs with large-scale optical dot products, rather than as a full engineering solution optimized for competitive performances, the computational speed and energy efficiency reported above should be interpreted as experimental conditions instead of as the specification of a final engineering product. For interested readers, a comparison of the computational speed and energy efficiency of several cutting-edge ONNs can be found in Ref. [18].

4.7 Discussion

Here we have presented experimental results that support the notion that optical neural networks can in principle [23, 15, 24, 25] have a fundamental energy

Photons (optical energy) per dot product				
Layer 1 → 2 (784-100)	Layer 2 → 3 (100-100)	Layer 3 → 4 (100-10)		Classification accuracy (%)
26.7 (10.1 aJ)	3.2 (1.2 aJ)	3.4 (1.3 aJ)		11.5
126.3 (47.8 aJ)	15.9 (6.0 aJ)	16.1 (6.1 aJ)		53.8
253.5 (95.9 aJ)	32.0 (12.1 aJ)	31.5 (11.9 aJ)		70.0
520.7 (197 aJ)	62.6 (23.7 aJ)	64.0 (24.2 aJ)		90.0
2439 (923 aJ)	314.5 (119 aJ)	306.6 (116 aJ)		99.2
Photons (optical energy) per scalar multiplication				
Layer 1 → 2 (784-100)	Layer 2 → 3 (100-100)	Layer 3 → 4 (100-10)	Weighted average	Classification accuracy (%)
0.034 (0.013 aJ)	0.032 (0.012 aJ)	0.035 (0.013 aJ)	0.034 (0.013 aJ)	11.5
0.161 (0.061 aJ)	0.159 (0.060 aJ)	0.161 (0.061 aJ)	0.16 (0.06 aJ)	53.8
0.323 (0.122 aJ)	0.319 (0.121 aJ)	0.316 (0.120 aJ)	0.32 (0.12 aJ)	70.0
0.665 (0.252 aJ)	0.627 (0.237 aJ)	0.638 (0.241 aJ)	0.66 (0.25 aJ)	90.0
3.109 (1.176 aJ)	3.144 (1.190 aJ)	3.070 (1.162 aJ)	3.11 (1.19 aJ)	99.2
Optical energy per inference				
Layer 1 → 2 (784-100)	Layer 2 → 3 (100-100)	Layer 3 → 4 (100-10)	Whole network	Classification accuracy (%)
1.00 fJ	0.12 fJ	0.013 fJ	1.14 fJ	11.5
4.83 fJ	0.61 fJ	0.067 fJ	5.50 fJ	53.8
9.68 fJ	1.22 fJ	0.13 fJ	11.0 fJ	70.0
19.9 fJ	2.40 fJ	0.27 fJ	22.6 fJ	90.0
93.2 fJ	12.0 fJ	1.28 fJ	107 fJ	99.2

Table 4.2: Breakdown of the average detected optical energy consumption, by layer, during the execution of the ONN for one inference

Table 4.3: Power consumption of each experimental device

Device	Power (W)
OLED display	~ 0.5
SLM + HDMI driver	18.5
MPPC detector	1.1
Digital laptop computer	~ 45
Total	~ 65

advantage over electronic neural-network implementations. We showed that ONNs can operate in a photon-budget regime in which the standard quantum limit (i.e., optical shot noise) governs the achievable accuracy. In particular, we achieved high classification accuracies using our ONN even when restricted to a photon budget of less than one photon per scalar multiplication. We used a standard neural-network model architecture and standard training techniques, so did not specialize the model to our hardware, nor did we need to perform any re-training to run the model on our hardware setup. This successful separation of software and hardware development shows that our ONN could be used as a drop-in replacement for other more conventional neural-network-accelerator hardware [11] without the need for any major changes to the machine-learning software workflow.

Our results have been enabled by several key design choices. The 2D-block design presented in this work, which can be seen as a generalization of the *Stanford matrix-vector multiplier* (Section 3.7) that avoids cylindrical lenses and their practical limitations, takes full advantage of the number of parallel modes in 3D free space, allowing extremely large numbers of scalar multiplications to be performed in parallel—up to ~0.5 million in our experimental realization. This

enabled us to implement a fully parallelized optical matrix-vector multiplier for matrices having size up to 711×711. The same optical setup could be, and was, used to compute vector-vector dot products with vectors of size up to ~0.5 million, which is many orders of magnitude larger than the vector sizes used in previous demonstrations of optical processors [17, 61, 56, 85]. The use of 2D blocks was also an important contributor to the achieved accuracy, since this layout reduces the impact of crosstalk between pixels (Section 3.7).

We have shown that the *optical* energy used to perform each scalar multiplication can be $<1 \times 10^{-18}$ J in a functional ONN performing MNIST handwritten-digit classification. The ONN demonstrated in this work was of moderate size, comprising layers with at most 784 neurons, and even better energy efficiency can be achieved for neural networks with wider layers [24]. Our results support an estimate that an ONN with appropriate system-level design could have an advantage of at least two orders-of-magnitude over electronic DNN accelerators, using a total energy across the entire system of only 1×10^{-16} J per scalar multiplication, even when operating at GHz speeds.

Our proof-of-principle results for sub-photon-per-multiplication ONN operation should readily translate to other ONN architectures, including those using coherent light, provided that they involve summation of a sufficiently large number of modes (be they spatial, frequency, or temporal modes) [24, 25].

One critical step towards building practical ONNs with high overall energy efficiency is to design scalable optical fan-out and fan-in using miniaturized passive components. Optical fan-out is crucial to realizing sub-photon scalar multiplications: an optical fan-out stage [64] would be used to spread photons from a single light source across a number of spatial modes larger than the number of

photons emitted (in the experiments reported in this paper, less than 1 photon per mode was achieved by applying attenuation, which, while enabling our scientific demonstration, is not suitable as an ultimate engineering solution). Both optical fan-out and optical fan-in can be implemented, for example, with lens arrays [76, 55]; demonstrating an ONN using highly scalable optical fan-out/in stages, integrated with high-efficiency modulators [86, 87] and detectors [88], is an important next step.

More broadly, the ability to perform low-precision matrix-vector multiplications more efficiently could find application beyond neural networks, including in other machine-learning algorithms [89] and for combinatorial-optimization heuristics [90, 91, 92].

CHAPTER 5

OPTICAL TRANSFORMERS

Much of the content in this chapter is adapted from the work presented in Anderson et al. [2]. My major contributions to this work included designing and performing the experiments, analyzing the data, and preparing the paper alongside the other co-authors.

In the previous chapter, we introduced an ONN implementation with very low optical energy. However, only the optical energy was considered. In this chapter, we further explore the potential overall energy advantages of ONN implementations for artificial intelligence models, particularly for very large models.

5.1 Background

In recent years, Transformer models [93] have emerged as a pivotal advancement in natural language processing (NLP) and beyond, revolutionizing the field with their ability to capture intricate dependencies and patterns in sequential data. The widespread excitement and unprecedented hype surrounding ChatGPT [94] underscore the transformative potential of these models, capturing the imagination of both the public and the scientific community. It has been found in particular that Transformer architectures significantly improve when sized up to billions or even trillions of parameters [95, 96, 97, 98, 99, 100, 101], causing an exponential growth of deep learning compute usage [102, 103]. These large-scale Transformers achieve ever more impressive results in not only natural language processing, but also in other domains such as computer vision [104, 105], graphs [106], and in multi-modal settings [107, 108, 109, 110, 111, 112],

making them a popular but expensive solution for many tasks—digital hardware's energy efficiency (ie. per-flop or per-inference cost) has not kept up with the growing FLOP requirements of state-of-the-art deep learning models [103]. They also have transfer learning capabilities [113, 114, 115, 95, 116, 104], allowing them to easily generalize to specific tasks, in some cases in a zero-shot setting where no further training is necessary [95, 110, 117].

Analog accelerators, such as optical accelerators, promise significant energy and speed advantages for neural network computation. However, due to their complexities and limitations, they have not yet been demonstrated with state-of-the-art large-scale artificial intelligence models. Consequently, the claimed energy advantages remain unproven. A Transformer model typically consists of millions of parameters, with their size rapidly increasing over the years (GPT-3 [95] has 175 billion, GPT-4 [101] has 1.76 trillion parameters). These figures far exceed the capabilities of current analog optical implementations. Previous work has considered deep learning models such as MLPs and convolutional networks on benchmark tasks like MNIST [63, 1], and simulations of convolutional models such as AlexNet [118] on more difficult datasets such as ImageNet [24]. This is important in understanding the viability of these systems for low-power and edge applications, but also begs the question of how well newer, larger models perform on optical systems. Here we study Transformers running on ONN hardware to understand the operation of ONNs at compute scales that are orders of magnitude larger than previously considered.

In this chapter, we use our free-space optical multiplier (as introduced in Chapter 3) to demonstrate Transformer operations in optical experiments and for our simulations. This system has many of the same characteristics as other

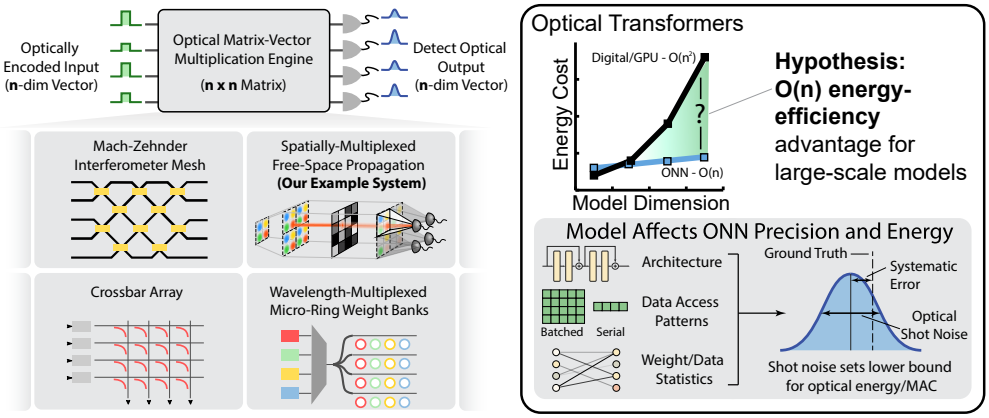


Figure 5.1: **Can Transformers benefit from running on optical hardware?** Left: Optical Neural Networks (ONNs) have been proposed as an alternative computing platform that can achieve asymptotic energy-efficiency advantages over digital computers running neural networks. There are various ONN platforms that all aim to efficiently implement matrix operations. Right: We hypothesize that Transformers' architecture allows for ONN-enabled benefits that scale. But energy-efficiency advantages with ONNs are not a guarantee; their behavior is affected by model architecture, statistics, and resilience to the noise/imprecision of analog hardware. Thus, while there are many implementations of general-purpose optical matrix accelerators (such as those depicted in the inset), there are still model-dependent challenges/tradeoffs in realizing their purported advantages. We seek here to answer the question of how much today's enormous Transformer models can benefit from this technology.

ONN implementations, and we aim to draw conclusions that could generally be useful for those working with other ONN designs (Section 2.2.2, see also Figure 5.1). While the individual implementation details are complex, what is most important are the shared traits: free data transport, analog noise/error (including optical shot noise), and linear operation computation (i.e. ability to perform matrix products). Using these high-level assumptions we aim to model the behavior and efficiency of Transformers running on ONNs in general.

5.1.1 Optical Neural Network Energy Calculation

Streaming Weights or Weights-In-Place There are two approaches for loading weights.*Weights-in-place* schemes involve loading them once, and re-using them for many inputs. Alternatively, systems can employ *streaming weights* where at every computation the required weight matrix is loaded. Streaming weights systems may be advantageous in situations where both operands of a matrix product are changing, such as in attention, or when weights are too large to be maintained by a weight-stationary device all at once; in such cases the weights would need to be reloaded for a weights-in-place system which are typically not optimized for doing so.

Estimating ONN System Energy Consumption ONNs' energy consumption is modelled as follows: the energy cost is broken down into the optical costs of performing MACs and the electrical costs of loading/detecting data, which are usually dominant. Consider a product between two matrices, $A \in \mathbb{R}^{n\times d}$, $B \in \mathbb{R}^{d\times k}$. Such a product results in loading (detecting) $nd+dk$ (nk) scalars, and performing ndk MACs. If the energy to electrically load (detect) a scalar is E_{load} (E_{det}), and to perform a MAC optically is E_{optical}, then the total energy is:

$$E = (nd + dk)E_{\text{load}} + nkE_{\text{det}} + ndkE_{\text{optical}} \tag{5.1}$$

For weights-in-place systems, one of the operands' loading costs can be assumed to be free, but in some systems maintenance of the weights which can be modelled as a small cost per MAC (given a certain throughput rate), E_{maintain}. Data access costs typically remain dominant due to the high costs of

DAC/ADC. The calculation is then as follows:

$$E = ndE_{\text{load}} + ndkE_{\text{maintain}} + nkE_{\text{det}} + ndkE_{\text{optical}}. \tag{5.2}$$

This illustrates how ONNs may have asymptotic energy advantages over digital computers. Notice that regardless of the number of reuses, all data is only loaded once in Equation 5.1 (and partial products are accumulated at a detector before converting and storing the data digitally). Meanwhile, E_{optical} ideally scales as $1/d$. These properties make energy cost disproportional to the number of MACs, ndk (assuming negligible E_{maintain} for 5.2, which it typically is, and in some architectures it is zero). In other words, $\frac{E_{\text{digital}}}{E_{\text{ONN}}} \sim \min(n, d)$. For weights-in-place operations, the energy advantage scales as $\frac{E_{\text{digital}}}{E_{\text{ONN}}} \sim d$ because the weights may be reused for free.

In general, many ONN accelerators share the same approach to data processing: data is read from memory, converted to optical signal, operated on by an optical system, converted back to digital, and stored. Estimates for energy costs follow the typical breakdown [1, 24, 119]: The energy E_{load} is broken down into three components, related to the energy of the cost of reading from memory E_{read}, digital-to-analog conversion (DAC) E_{DAC}, and modulation to generate the light E_{mod}:

$$E_{\text{load}} = E_{\text{read}} + E_{\text{DAC}} + E_{\text{mod}}. \tag{5.3}$$

Detection energy consumption E_{det} can broken down in a similar fashion, where

$$E_{\text{det}} = E_{\text{detector}} + E_{\text{amp}} + E_{\text{ADC}} + E_{\text{write}} \tag{5.4}$$

represent the costs of detecting a signal, amplifying the detected signal, performing analog-to-digital conversion, and writing to memory respectively. Often, the goal is to amortize these data-access-related costs via the data reuse when computing with large operands.

Importance of Neural Network Architecture In practice, achieving an efficiency advantage with ONNs is dependent on the neural network architecture being run. Data access (storage) from digital memory occurs before (after) digital-analog (analog-digital) conversion, so the costs for loading/storing from digital hardware are only paid once, since the optically-encoded data is freely transported/copied. Thus, the more compute to be performed per data access (ie. how large matrix-vector products are, which depends on the DNN architecture), the more efficient the ONN. Also, the precision requirement must not change when the model being run is scaled. If this is true, then under shot noise, only the same mean number of photons at the output of a dot product is necessary, no matter how large the computation is.

5.1.2 Quantization of Large Language Models (LLMs)

Optical hardware's low precision raises the question of whether scaled-up models could be quantized sufficiently to run. Thankfully, continual research in LLM compression has progressively shown that larger models do not have increasing precision requirements. For example, [120] found that larger Transformers can be compressed more easily, to the degree that it is more worthwhile to train large ones and compress them over training smaller ones of the target size. Furthermore, [121] and [122] demonstrated running Transformers at scale with int8

precision, and the recent work of [123] proposes that 4-bit is optimal for nearly all model scales, except for the largest tested (175B parameters) where 3-bit was sometimes found to work better. Some approaches utilize quantization-aware training [124](QAT), where a model is fine-tuned while subject to quantization, to make it robust at low precision.

5.2 Overview

Here we demonstrate how Transformers run on ONN systems, and estimate the potential benefits of doing so. To first verify that Transformers may run on these systems despite their imprecision, we sampled operations from a Transformer and ran them on a real spatial light modulator (SLM) based experimental system, and used the results to create a calibrated simulation of the optical hardware, with the systematic error, noise, and imprecision of weights/inputs we observed. Transformers running on the simulated hardware could perform nearly as well as those running digitally, and could be far more efficient. We summarize our key contributions as follows:

- We demonstrated linear Transformer operations (the bulk of a Transformer's computation) running with sufficient accuracy on real optical hardware and in a matching simulation, despite errors and noise on hardware supporting fewer than 8 effective bits of precision.
- Via simulation, we established scaling laws for optical Transformer performance versus optical energy usage, and optical energy usage versus model size. We found that Transformers accelerated optically achieve performance that is consistent with the ideal $\frac{1}{d}$-energy-per-MAC scaling

possible on analog hardware, and that Transformer architectures are large enough to benefit significantly.

- Based on our simulations and experiments we estimated an orders-of-magnitude energy consumption advantage of full ONN accelerators versus state-of-the-art GPUs, exceeding 10^3 for near-future model sizes.
- We discussed how Transformers' suitability for optical acceleration is related to their architecture, and more generally how specific elements of DNN architecture affect the function of ONN systems running them.
- We identified the hardware and systems design challenges that future work on building ONN accelerators should target.

While our experiments and simulations were based on specific hardware as a representative example, our scope here is more general. We are interested in understanding how optical energy scaling and noise relate to Transformer performance and architecture. As such nearly all our findings apply broadly to linear optical processors (and hopefully future ones), irrespective of their underlying hardware implementation details (Figure 5.1, left).

5.3 Optical Transformers

Transformers are models for processing sequential data based on multi-head attention. Transformers consist of two-layer feed-forward blocks and multi-head attention (5.2) operations. Multi-head attention computes relationships between sequence elements by deriving query, key, and value sequences Q, K, V and computing dot products with a softmax nonlinearity in-between [93]. Transformers

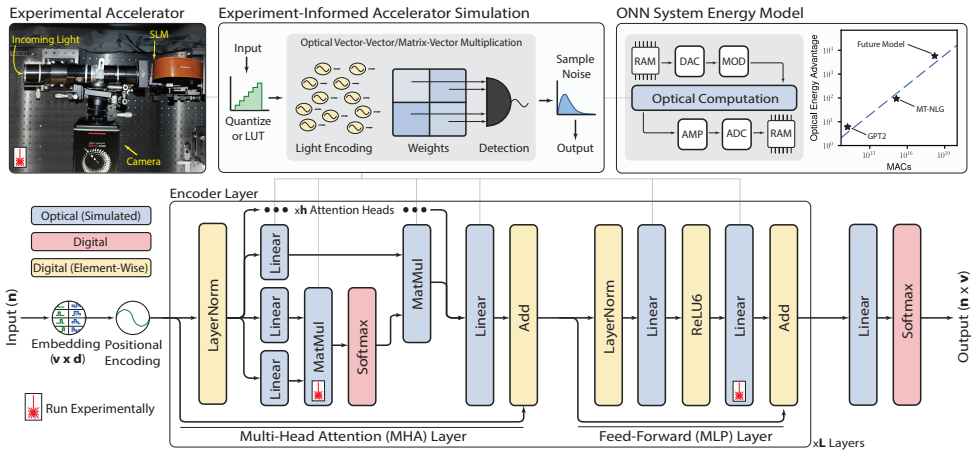


Figure 5.2: **Optical Transformer evaluation: prototype hardware; simulator model; Transformer architecture.** Bottom: typical Transformer architecture, but with ReLU6 activation. Top Left: experimental spatial light modulator (SLM)-based accelerator setup. From some layers—marked with a laser icon—we sampled dot products to run on real hardware. Top Middle: Linear operations, in light blue, run on a simulated accelerator with noise/error. Lookup tables (LUT) allow simulation using our setup's supported weight/activation values. Top right: our model of energy consumption for optical accelerators, based on assumptions and results from our experiment/simulations. The model accelerator system consists of random-access memory (RAM), a analog/digital conversion (DAC/ADC), light modulation (MOD), amplification (AMP).

also leverage modern design elements such as additive residual skip connections [125] and normalization layers [126]. A defining feature of Transformers is that entire sequences may be processed in matrix-matrix products in parallel (instead of one token/input at a time).

We designed models that are intentionally similar to other Transformers, with the goal of simulating their behavior (informed by some experimental measurements) and energy consumption on optical hardware. A summary of our approach and model is in Figure 5.2.

5.3.1 Architecture and Task

We created optical Transformer models with a GPT2-like [115] architecture that replaces the GELU [127] activation with ReLU6, which is known to improve low-precision model performance [128, 129, 130]. For language modelling, we used the raw Wikitext-103 dataset [131]. The models we simulated have 12 layers (consisting of multi-head attention and feed-forward blocks), operate on a context length of 1024 tokens, use 12 attention heads, and have embedding dimension d varying from 192 to 1536.

5.3.2 Transformer Computations on Optical Hardware

We ran experiments using a real Transformer's (we used the base-sized model with $d = 768$) weights in order to characterize the behavior of an ONN system. We adopted as a representative example of an optical accelerator a spatial light modulator (SLM) based system which computes vector-vector dot products [1]. Vectors are encoded on a display, and copies are shone through the SLM which has varying transmission corresponding to some data (ie. a weight matrix). The outputs of this operation—element-wise products—are collected at detectors as the resultant dot products (Figure 5.2, top left). We then collected calibration curves, mappings from the detected output light intensity to the actual neuron floating-point values. To do this, we ran many random dot products on the hardware and collected pairs of detected values and digitally-computed ground-truth values. We then fit the relationship linearly. We used high photon counts and averages over repeated experiments to eliminate shot noise, leaving any deviation from the linear fit as the hardware's *systematic error*.

There are differences between the precision limitations of real devices and linearly-spaced quantization schemes often used for DNNs - While these devices are commonly controlled by digital signals with evenly spaced discrete levels, the resultant output of these devices tends to be unevenly spaced because of their intrinsic nonlinear response or finite extinction ratios. We used lookup tables (LUT)s to model this kind of hardware error that is common to many optoelectronic devices. The LUTs were collected for the organic LED display and spatial light modulators (SLMs) by measuring levels of one device with the other at full transmission/emission. We incorporate these LUTs into both training and simulation. Backpropagation is carried out using the straight-through estimator just as for QAT, but unlike QAT once the rounding operation produces the quantized uint8 representations, the numbers are directly used to index the LUTs to produce the activations instead of dequantizing.

5.3.3 Simulation of Optical Hardware

Informed by our experiments, we constructed a simulation of the optical hardware. By simulating the hardware behavior directly we model how any arbitrary operation would behave if run on the physical setup when it is infeasible to run large models experimentally. We aimed to emulate the noise, error, and precision that we observed in order to understand how well full Transformers would perform when running on optical hardware.

Hybrid Scheme Pure optical systems cannot easily compute activation or normalization functions. Thus we assumed LayerNorm, ReLU activations, and residual skip connections are performed digitally at full precision. Thankfully,

even in smaller models, linear computations are the overwhelming majority (Section 5.4.3).

Non-Negative Weights and Inputs ("4-Pass" Multiplication) An important limitation is that our display and SLM only support non-negative values. The constraint of having all-positive data is present in many but not all optical neural network systems.We worked around this by decomposing products into sums/differences of products with non-negative operands. Consider a product between matrices W and X. If we let W_+ (X_+) and W_- (X_-) be matrices with only the positive and negative elements of W (X) respectively, then:

$$WX = W_+X_+ - |W_-|X_+ - W_+|X_-| + W_-X_- \tag{5.5}$$

Data Scaling On the real system, we define a maximum activation/weight value as 1.0 and minimum as 0.0. To simulate operation, the inputs and weights of every simulated NN layer are scaled to this range, and then rescaled back afterwards.

Device Quantization Real hardware may only have certain number of representable levels. To emulate this behavior, we fine-tuned pretrained models using QAT and applied the following in simulation:

- For optics-simulated layers, we applied quantization to int8 (256 levels). Then, instead of dequantizing, we used the integer values directly as indices into the LUTs that we gathered experimentally.

- We also quantized weights, but with the SLM LUT. We clamped smaller values to 0.02 in the simulation, as our SLM does not have a high extinction ratio, and the smallest transmission is 0.02.
- Accumulation can be high precision, but we used int8 quantization for outputs, since analog-digital conversion (ADC) is expensive in practice.
- We used both deterministic and stochastic rounding when quantizing, with similar results.

Systematic Errors Issues like cross-talk, misalignment, defects in ONNs give rise to systematic errors. We simulated such a constraint by adding Gaussian noise to simulated model outputs, scaled relative to the mean sizes of the outputs, as this was the noise behavior we observed experimentally (it is related to the rescaling of data between 0 and 1).

Optical Encoding and Shot Noise We modeled optical encoding by subjecting layer outputs to simulated shot noise (Figure 5.2), which differs from the systematic error model. Outputs were scaled by a number such that the average photon number per feature (photons/MAC) was some target value. Each of these features was used as the mean of a Poisson distribution, which we sampled. These outputs were then scaled back down to represent neuron values. In the simulations for optical scaling we used vanilla 8-bit QAT (no LUTs or systematic error, which can overwhelm shot noise) to cleanly demonstrate the optical scaling properties—which are model-dependent and not hardware-dependent—of Transformers.

5.4 Results

5.4.1 Transformer Error Tolerance and Simulation Accuracy

We determined experimentally that Transformer operations are able to run on real hardware without severely degraded performance from systematic errors. The bottom four panels of Figure 5.3 are histograms of the experimental differences from correct values. The simulated noise distributions (dotted lines) match well with the experimental data, which confirms that they are an accurate representation of the real systematic error behavior. Figure 5.3 (top) is a map of the performance of the simulated model over different configurations of the mean-relative (in percent) noise at every layer of feed-forward and attention blocks. The model performs well with significant noise (experimental noise levels marked with stars), within 1 perplexity from noise-free performance unless the noise is very high. While 8-bit precision was used for QAT, the optical Transformer can perform inference at lower precision, as implied by its error tolerance.

5.4.2 Optical Scaling Laws

Optical Transformers achieve language modelling performance close to their digital counterparts' when shot-noise-limited at photon budgets where optical energy is negligible. The perplexities on the Wikitext-103 validation set of various optical Transformer models simulated with different total photon usage (amount used for input data) are shown in Figure 5.4 (left). The curves illustrate a tradeoff: larger models need larger photon totals to function well, and there

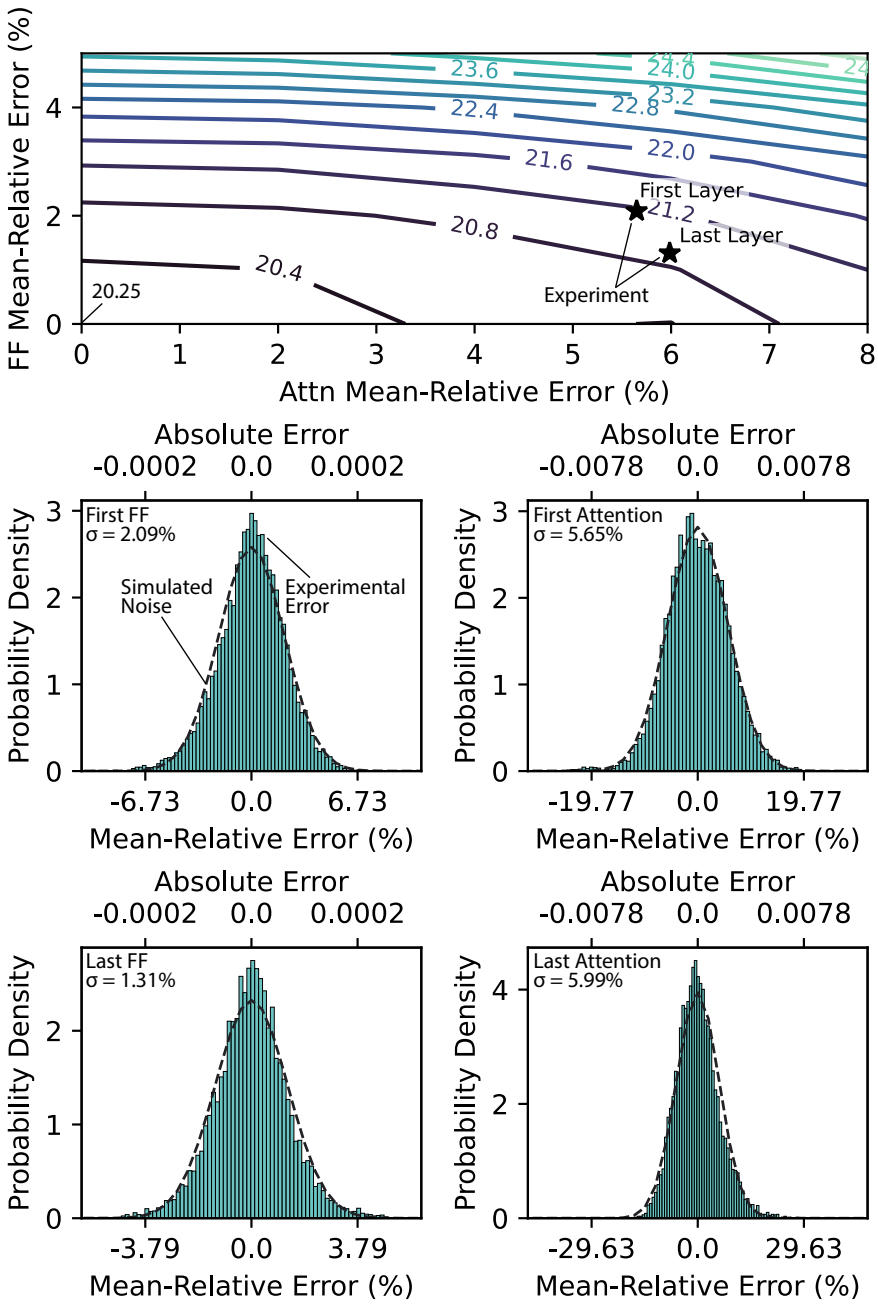


Figure 5.3: **Comparison of experimental and simulated noise models and simulated Optical Transformer noise tolerance.** Top: Simulated performance (Wikitext-103 validation perplexity (PPL)) versus percent mean-relative simulated noise in feed-forward (FF) and attention (Attn) layers. Systematic errors from experimental data marked with a star. Bottom: comparison of simulated noise model to error from experimental data. The Gaussian shape of the simulated error behavior matches experiment accurately.

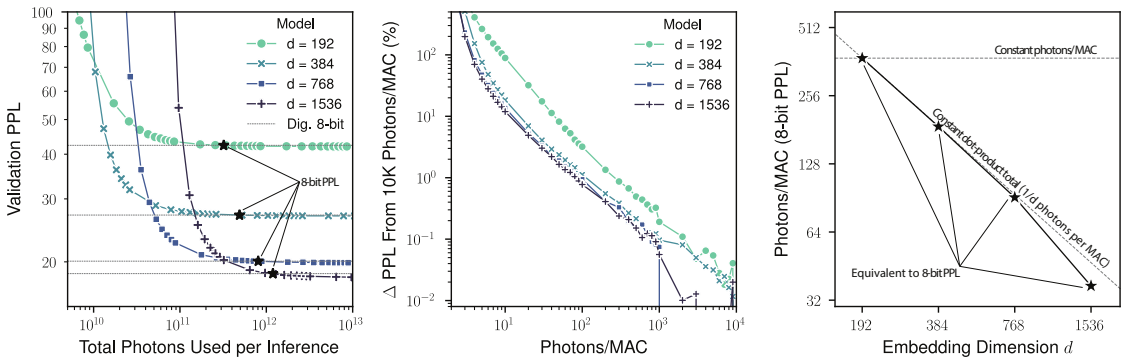


Figure 5.4: **Simulations of Optical Transformer behavior with varying photon usage.** Left: Wikitext-103 validation-set perplexity (PPL) versus embedding dimension d and total photons used for a single forward pass/inference. 8-bit digital model performance is shown with dashed lines. Middle: perplexity degrades from ideal with fewer photons-per-MAC; the plot exhibits truncated power-law scaling. Right: Scaling of number of photons needed for an Optical Transformer to achieve the same perplexity as an 8-bit digital-electronic processor, versus model size.

are different optimal model choices based on the photon budget. We define photons/MAC as the total photon budget (amount at input) divided by total MACs. The percentage difference from the performance at 10K photons/MAC (Figure 5.4, middle)—chosen to represent an ideal high-precision scenario—is roughly power-law scaled in photons/MAC for all models with truncation near 10K; better performance can be had with more photons, but with diminishing returns, and the performance matches or exceeds that of the 8-bit digital models' when the photon budget is not too low ($\sim 10^2$).

The models use fewer photons/MAC as they scale, achieving the theoretical efficient scaling where the total per-dot-product photons needed is constant. To study how photon usage scales, we determined how many photons it takes to reach the performance of 8-bit digital models. These values, in Figure 5.4 (right), decrease nearly as $\frac{1}{d}$—the total photons needed per dot product is con-

stant (bottom dashed line). The Transformer architecture clearly takes advantage of efficient optical scaling with larger model sizes, suggesting that required output SNR does not increase with scale. This is consistent with other work which found that Transformers compress/quantize well at scale [120]. Meanwhile, the already low photon usage of the largest model suggests that models larger than our simulations (¿10B parameters) may use ¡1 photon/MAC. This sub-photon operation works in optical systems [1, 21] and is in essence no different at all from operation at higher photon counts (since the number summed at detection is still high). These empirical scaling results are tied to our specific configurations and training strategies.

5.4.3 Estimated Energy Usage

The efficient photon scaling trend we observed in Section 5.4.2 suggests that Transformers running on optical hardware could achieve significant energy efficiency advantages over running on digital hardware. To understand the efficiency of Transformers on optical hardware, we designed an ONN system based on current hardware that is like our experimental setup, with our measured precision and photon scaling (see: Figure 5.2, top right). It is an inference system with in-place weights which are loaded once and reused forever, activations read from and written to SRAM for every layer, a 10 GHz light modulator array, and an optical "core" which can perform 10M multiplications per cycle (this can be thought of as a 10 megapixel SLM). The photon-per-MAC scaling versus model dimension is taken to be the $1/d$ scaling which we found was possible in our simulations, and we assumed that the model operates with 5-bit input precision, 8-bit weight precision, and 7-bit output precision, as determined by

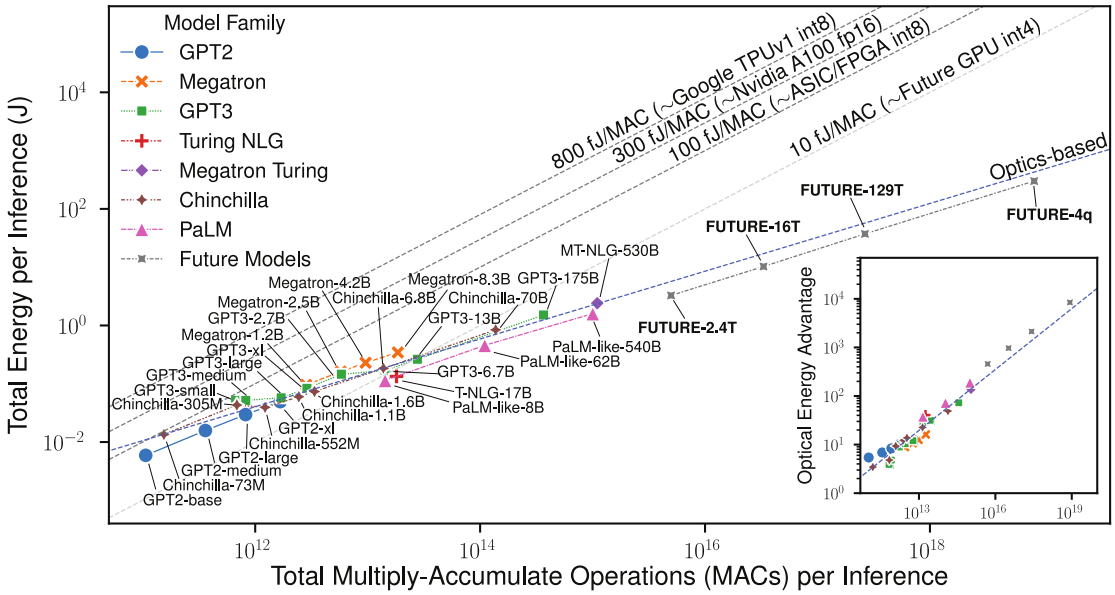


Figure 5.5: **Estimated energy usage of Transformer models on optical hardware for a single forward pass/inference.** Hypothetical future model designs are labelled **FUTURE-***. Trend for energy usage in optical systems (blue) computed based on real models only. Inset: energy advantage of running on optics over estimated NVIDIA A100 usage. The advantage grows with the model compute. M = 10^6, G = 10^9, T = 10^{12}, q = 10^{15} parameters.

our study of low precision performance.

Our approach to energy estimation is as follows: the system is thus assumed to have the behavior and components as described in Section 5.1.1; we use Equations 5.1 and 5.2 to calculate the energy cost for every linear operation in Transformer models, with $E_{\text{optical}} \sim 1/d$. This includes both the model's linear layers (Equation 5.2, as weights are assumed to be kept in place) and the matrix-matrix products among activations in the attention operation (Equation 5.1, which includes the cost of loading *both* operands, as there are no static weights). The cost of digitally-run operations, such as softmax, ReLU, and other element-wise operations, is assumed to be the cost of storing and loading all operands in mem-

ory. None of the digital operations have the $\sim d^2$ scaling of the optically-run linear operations, so energy-efficiency advantages are still possible.

As models grow, running Transformers on optical hardware has a large and asymptotic efficiency advantage over running on digital hardware. In Figure 5.5 we chart estimates of the forward pass energy required for various models[1], including a hypothetical family of large, dense Transformer models designed in a similar fashion, which we label **FUTURE-***. For comparison, we also chart various digital systems [27] in different performance regimes, and a hypothetical "next generation" GPU that can use ~10 fJ/MAC. For small models, the optics-based system uses about the same energy, but eventually gains an advantage that scales asymptotically with the number of MACs. For the larger models, MT-NLG-530B and FUTURE-4q, the optics-based approach would have ~140× and ~8500× energy advantages over the current state-of-the-art GPU (NVIDIA A100) respectively.

In summary, we found that as models get larger the feed-forward layers require most of the computation, but that the energy of data access in attention is still very expensive due to the detection of many attention matrices across heads. Meanwhile, the costs of the digital operations become relatively small, $\leq 20\%$ of the total energy for large models, and therefore not a significant bottleneck. [2]

[1]The recent PaLM [132] models used a modified architecture. For simpler comparison, we make our estimates using a model with GPT-like architecture but with the PaLM model dimensions, which we call PaLM-Like.

[2]Trends in the design of real models have increasingly favored optics over time. Specifically, attention loads/stores a $n \times n$ attention matrix for each of the h attention heads. Models with more MLP compute per attention head have a larger overall ratio of computation to energy usage; larger $\frac{d}{h}$ is more efficient. The largest GPT2 [115] uses $\frac{d}{h} = 64$; GPT3 [95], 128; MT-NLG-530b [133], 160; and PaLM [132], 384.

5.5 Discussion

The results given in Section 5.4.3 on optical Transformers' efficiency have implications for the design of future ONN hardware/software systems. In particular, we found that ONNs constructed with the following traits would be ideal:

- An efficient ONN system for Transformers must perform data copying after digital-analog conversion (fan-out) and accumulation (fan-in) of partial products before detection and analog-digital conversion.
- An ONN must perform computations with large operands in a single shot to gain an energy advantage. Once operands exceed $10^4 \times 10^4$ in size the advantage is significant, and therefore a future ONN should implement at least this level of parallelism to achieve >100× efficiency improvements over current state-of-the-art GPUs (NVIDIA A100).
- An ONN must support at least 7 effective bits of precision. The imprecision can come from various sources of noise or error as long as there are effectively $\sim 2^7$ distinguishable levels (however, recent work has demonstrated low-precision Transformers [123, 134]).
- An ONN system must have sufficient fast (ie. SRAM) memory to store activations at minimum.
- An ONN should be implemented with minimal cost of its surrounding electronic components for maximum benefit. Future improvements in CMOS technology will be greatly beneficial. We estimate that future optics-based systems might achieve energy advantages of $>100,000\times$ running models the size of FUTURE-4q (over 300 fJ/MAC).

These generic traits/specifications are sufficient to implement Transformers efficiently, regardless of implementation details. But the design of the software — DNN architectures, including Transformer shape and size — for these systems is also critical:

- The asymptotic advantage of optics is that once data is loaded, it may be reused N times for free, with constant energy for M-sized dot products. This suggests that architectures with large M and N benefit the most, and that wider is better than deeper when scaling a model (in terms of energy).
- The attention mechanism requires much of optical Transformers' power consumption for very little compute. Models designed with larger $\frac{d}{h}$ are therefore more energy efficient. Scaling of Transformers is conveniently following this trend already.
- The ability of Transformers to run efficiently optically is due to their parallel-processing of tokens with the same weights, and ability to tolerate the levels of noise and error present in ONN systems at scale. Thus architectures designed with similar behavior [135, 136] could also be efficient.

Limitations While we have laid out the potential requirements and used simulation to predict the viability and potential gains from doing so, building a full ONN system that realizes the potential benefit is still an open challenge. For example, while optical components may perform computations cheaply and quickly, there is still the issue of supplying them with sufficient data bandwidth to fully leverage them. Furthermore, studying potential advantages in speed/throughput is more challenging. In this domain, different ONN implementations have different behavior, and may be optimized towards running

certain neural-network architectures, and further study of the tradeoffs with speed/energy/memory could be necessary. Integrating these components into a working system – memory, conversion, the optical elements, etc. – presents a significant engineering challenge. Finally, we note that while our estimates are for single devices, large-scale deep learning systems often consist of multiple devices working together, due to memory/compute limitations. This introduces additional data-transport costs for digital systems, and memory-bound situations affect both digital and optical systems' energy consumption in non-trivial ways. In these cases, more complicated schemes to run models efficiently are necessary, such as sharding weights across devices, and the assumption that all weights can be kept in-place must be relaxed. Despite these limitations, we hope that the potential benefits we studied here motivate future work in this direction.

5.6 Summary and Conclusion

We have demonstrated the ability of Transformer models to run accurately and efficiently on optical hardware through optical experiments and an experiment-informed simulation of the hardware. We examined Transformers' scaling behavior with optics and used our findings to show that optical systems have a large and asymptotic energy advantage over digital ones that *grows* with the model size. For example, we showed that optical hardware may achieve an over 100× energy advantage when running the largest Transformer models today (~500 billion parameters) and that larger, future Transformers (~4 quadrillion parameters) may be realized with an >8000× optical energy advantage. We then explained how existing trends in the machine-learning community (such as op-

timizing Transformer designs for better parallelism, and building models that rely more heavily on multi-layer perceptrons within them [135]) are resulting in models that are increasingly better suited for optical hardware than the canonical Transformer model [93] and previous language models—they involve large, dense, linear calculations to saturate processors' compute capabilities without data-access bottlenecks. This is not a coincidence; in general, neural-network models that are designed to be more efficient on digital-electronic hardware by maximizing compute and parallelism per-data-access will be more efficient on optical hardware, where data-access costs are also a primary concern.

We discussed what specifications optical hardware would need to meet to realize the projected energy-efficiency advantages for different model sizes. For example, to realize the 100× energy-efficiency advantage for current Transformer models, we project needing optical matrix-vector multipliers each capable of multiplying a $10^4 \times 10^4$-dimensional matrix with a 10^4-dimensional vector at a rate of 1 matrix-vector multiplication every 0.1 nanoseconds, with power consumption of roughly ~1 pJ/bit for input/output data access and conversion. While existing components are capable of this level of energy-efficiency, there is an open hardware systems-level challenge of making an optical neural network that is simultaneously big, fast, and efficient enough. We also discussed how future architectural changes and improvements to electronics would further improve ONN efficiency—with better model quantization and assumptions about near-future hardware, future Transformers running on optical hardware could exceed a 100,000× energy advantage over current state-of-the-art ~300-fJ/MAC digital electronics. The efficiency of digital-electronic neural-network accelerators is improving at a rate of ~10× every 7 years [137], suggesting that even if the proposed optical hardware takes 10 years to develop, there should still be

a several-orders-of-magnitude benefit to using optics for neural-network computations when the scaled hardware becomes available. Furthermore, any improvements to the energy efficiency of electronic memories directly benefits both digital processors and optical processors, so future development of digital processors may only shrink the optics–electronics efficiency gap through improved MAC-computation efficiency, which has proceeded predictably for the past two decades based on transistor scaling [138].

We believe our findings about the potential energy-efficiency of optical accelerator hardware strongly motivate pursuing the development of optical processors for large-scale deep learning with Transformers (or other models that heavily rely on weight-stationary matrix-vector multiplication).

CHAPTER 6

MULTIMODE PHOTON COUNTING FOR QUANTUM OPTICAL STATES

Much of the content in this chapter is adapted from the work presented in Presutti et al. [3]. My major contributions to this work included evaluating and calibrating photon-counting cameras for the spectrometer, collaborating on the quantum-optical modeling of the experiment with the other co-authors, and assisting in building and running the experiment.

6.1 Background

The generation, control, and measurement of entangled multimode Gaussian states of light are crucial elements of continuous-variable (CV) quantum information processing [139, 140, 141, 142]. Most quantum technologies based on multimode quantum optics benefit from being able to use as many modes as possible. As an example, a Gaussian boson sampler (GBS) [143, 144] is a special-purpose quantum computer that can – at least in the ideal case – perform certain calculations that are believed to be intractable on a classical computer when the number of modes and number of photons in the GBS are sufficiently large [144, 145]. The recent demonstration of Gaussian boson sampling in the regime of quantum computational supremacy, with tens to hundreds of squeezed modes and detected photons per shot [146, 147, 148, 149], was a milestone in the development of CV-based quantum systems that was achieved because of the success in pushing to large numbers of modes and photons. A GBS executes a sequence of three steps, which are common to many CV quantum-information-processing protocols: (1) generate squeezed states, (2) apply a unitary transformation to entangle them, and (3) measure the final state (by photon

counting).

Optics gives us the choice of several possible degrees of freedom within which to encode information – most importantly: space, time and frequency (or any combination thereof). While large-scale GBS experiments have been realized using space [146, 147, 148] and time [149] encodings, the frequency domain remains to be explored. Frequency encoding offers important potential advantages over space or time encoding for both the generation and the manipulation (unitary control) of multimode squeezed light: reduced hardware resources and complexity, and reduced loss. The extremely broad bandwidth of light enables frequency-encoded systems to operate on many frequency modes in a compact system [150, 151, 152, 153, 154, 155, 156]. Many demonstrations of large-scale multimode squeezing, for example, use the frequency domain, e.g., in broadband frequency combs [154, 155, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169]. Linear unitary operations in the frequency domain, i.e., acting on the frequency modes, can be implemented in a hardware-efficient way, operating on all frequency modes in parallel [170, 171]. One approach is to use one or more electro-optic modulators [172, 173, 170, 171, 174] (although there are limitations on the unitary from the driving microwave bandwidth being $\lesssim$100 GHz); another is to use nonlinear wave mixing to convert photons in each frequency mode to photons in a combination of other frequency modes [175]. Unitaries based on nonlinear wave mixing mediated by a classical field, such as four-wave mixing [176], provide a route to realizing programmable unitaries that can operate over wide bandwidths, in compact hardware with low loss. The programming of the unitary can be done by shaping the classical field(s) used to control the wave mixing of the modes containing quantum light, and the wave mixing can be implemented compactly in a single-spatial-mode waveg-

uide. However, a programmable, frequency-domain unitary working over optical (>1 THz) bandwidths for quantum light has yet to be realized at even moderate scale (more than 2 modes [176]).

The final step, state measurement by photon counting, is a challenge for many multimode architectures. Since the preferred nonlinear optical materials for generating squeezed vacuum work best at longer wavelengths, squeezing at wavelengths centered near 1550 nm is typical [146, 147, 148, 149, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 169], also in part due to the convenience of being able to use optical components from telecommunications technologies. However, good (high quantum-efficiency, low dark-count) single-photon detectors at 1550 nm, namely superconducting nanowire detectors [177, 178], are very expensive and require cryogenic cooling. Silicon-based camera sensor technologies – both charge-coupled device (CCD) and complementary metal–oxide–semiconductor (CMOS) detectors – are well-established, comparatively inexpensive and compact, and each camera comprises 10^5–10^6 individual pixels, i.e., detectors. Cameras capable of detecting single photons with low noise have recently become available, and there is a growing literature of quantum-optics and sensing experiments that were performed with visible wavelengths and these cameras [179, 180, 181, 182, 183, 184]. Electron-multiplying CCD (EMCCD) cameras are arguably the current state-of-the-art, and low-noise CMOS photon-number-resolving cameras are also a promising tool within this domain [185, 186].

6.1.1 Theory of EMCCD Camera Statistics

The electron-multiplying (EM) gain process used in EMCCD cameras is what makes these instruments highly sensitive: by magnifying the charge of a few photoelectrons to macroscopic levels, the camera is capable of detecting single photons. However, the nature of the gain process is stochastic, which introduces noise. Here we introduce the model for the gain process, and discuss how it influences the measurements we are interested in.

Photon Statistics after EM Gain

The EM gain process adds a high amount of noise, which typically precludes the possibility of resolving the exact photon numbers in EMCCD cameras. When n photons are captured as photoelectrons on the CCD sensor and amplified with EM gain g, the conditional probability of yielding x amplified electrons follows a gamma (Erlang) distribution [187]:

$$P(x;n) = \frac{x^{n-1}\exp(-x/g)}{g^n(n-1)!}.$$

To estimate the statistics of the EM gain output x, let P_n be the photon-number distribution incident on a pixel. We are first able to calculate the conditional moments of x with respect to n, use these to calculate (unconditional)

moments of x, and finally calculate the photon number moments:

$$\begin{aligned}
\langle x^k \rangle_n &= \int_0^\infty x^k P(x;n) dx = g^k \frac{(n+k-1)!}{(n-1)!}, \\
\langle x^k \rangle &= \sum_{n \geq 0} P_n \langle x^k \rangle_n, \\
g^{-k} \langle x^k \rangle &= \sum_{n \geq 0} P_n \prod_{i=0}^{k-1} n + i = \left\langle \prod_{i=0}^{k-1} n + i \right\rangle.
\end{aligned}$$

This can be extended to correlations between multiple variables, e.g.:

$$\langle x_i^k x_j^l \rangle = \sum_{n_i, n_j \geq 0} P_{n_i n_j} \langle x_i^k \rangle_{n_i} \langle x_j^l \rangle_{n_j} = g^{k+l} \left\langle \left(\prod_{m=0}^{k-1} n_i + m \right) \left(\prod_{m=0}^{l-1} n_j + m \right) \right\rangle.$$

We may then solve for the photon number moments by inverting the above equations. Hence we can write down the formulae for our photon statistics of interest (omitting the gain term g):

$$\begin{aligned}
\langle n \rangle &= \langle x \rangle \\
\langle n_i n_j \rangle &= \langle x_i x_j \rangle \\
\langle n_i n_j n_k \rangle &= \langle x_i x_j x_k \rangle \\
\langle n^2 \rangle &= \langle x^2 \rangle - \langle x \rangle \\
\langle n_i^2 n_j \rangle &= \langle x_i^2 x_j \rangle - \langle x_i x_j \rangle \\
\langle n_i^2 n_j^2 \rangle &= \langle x_i^2 x_j^2 \rangle - \langle x_i^2 x_j \rangle - \langle x_i x_j^2 \rangle + \langle x_i x_j \rangle.
\end{aligned}$$

Finally, the electronic stages between the EM gain process and a digitized pixel value will introduce additional noise, such as readout noise. However this noise is independent of the signal x, hence it does not affect the above our

ability to estimate the photon number.

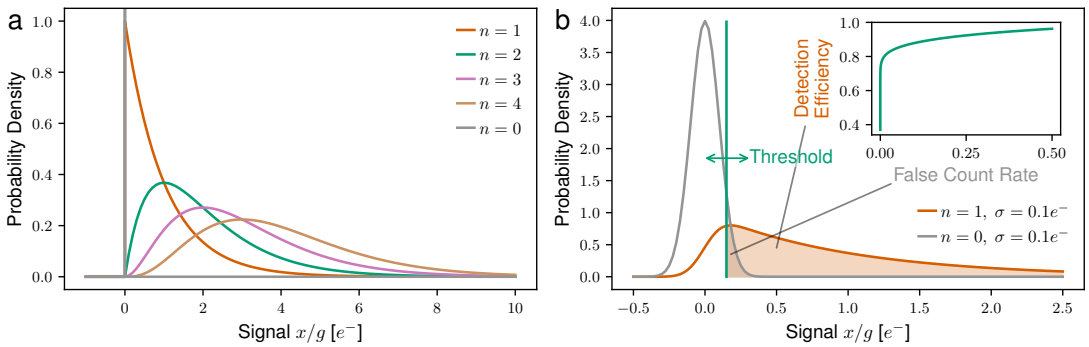


Figure 6.1: **Electron multiplying gain model and thresholding. a.** The probability distribution of an amplified signal for a given number of photoelectrons, prior to additional noise. We model this with the Erlang distribution. The signal x is measured in electrons (e^-) divided by the gain g (the expected value of this is the original number of photelectrons). **b.** The probability distribution of the signal with Gaussian readout noise. Although the two probability distributions begin to overlap, in this example, thresholding can be used to distinguish between 0 and 1 (or more) photons with some degree confidence, incurring a tradeoff between the photon-detection efficiency and false count rate. The readout noise has standard deviation σ. The inset shows the analytical (ideal) ROC curve for this example.

Thresholded Operation

As discussed above, the electronic signal x, ensuing an EM process with a gain of g, follows a random distribution parametrized by the number of photons captured n – this is shown in Figure 6.1. Due to the stochastic nature of the EM gain process, it is impossible to resolve the original number of photoelectrons n on the CCD sensor from the number of amplified electrons x. However, if there are no photons absorbed by the CCD sensor during the detection window, there is no signal (no electrons) to amplify in the gain process. Hence, there is no

excess noise from the EM process, and the variance of the output signal depends solely on the readout noise σ. With a high EM gain, the effective readout noise, $\sigma \ll 1\, e^-$, we can set a threshold on the output signal to discriminate between the absence or presence of photoelectrons with a high degree of confidence.

We evaluate the EMCCD camera in the single-photon detection mode using a threshold to detect the absence or presence of a photon. The threshold value will determine the photon-detection efficiency (PDE) and the false click rate. In general, a lower threshold increases the effective PDE but increases the false click rate, and vice versa. See Figure 6.1b. By varying the threshold value, we obtain a receiver operating characteristic (ROC) curve to characterize the performance of the photon counter (Figure 6.1b inset). The ROC curve quantifies the trade-off in the effective quantum efficiency and dark count rate. In practice, we obtain the false click rate as a function of the threshold by obtaining dark frames – images from the sensor with no illumination – and count how many times a given threshold is exceeded. We obtain the PDE by determining the readout noise from this data, and computing the PDE through the model by comparing g and e, multiplied by the quantum efficiency (QE) of the CCD sensor.

6.1.2 Photon Statistics in Multimode Gaussian Quantum Optics

The aim of this section is to provide a working knowledge of the physics of multimode Gaussian quantum optics, especially as it relates to our experiments. We provide an explanation of the underlying theory in our preferred formalism, show how to apply the nonlinear optics equations, and how to predict the

supermodes and squeezing values. Finally, we describe the photon number properties of Gaussian states.

We refer to as "Gaussian" the class of states that are fully described by the first two moments of its field operators, or, equivalently, whose phase space distribution is a Gaussian. These states are the most typical in experimental quantum optics (this is a consequence of weak optical nonlinearities and the large – classical – driving fields that are required to achieve them). Furthermore, while multimode or multivariable quantum-mechanical states have generally exponentially large representations, Gaussian states have efficient phase-space representations. This makes working with large multimode Gaussian states tractable.

The Bosonic Covariance Matrix

To fully characterize a zero-mean Gaussian state, we must know the squeezing, the thermal noise, the loss and the correlations between all modes; all this information is encoded in the covariance matrix. In typical convention [188, 189]:

$$\sigma = \frac{1}{2}\left\langle\left\{\hat{\xi}, \hat{\xi}^\dagger\right\}\right\rangle - \left\langle\hat{\xi}\right\rangle\left\langle\hat{\xi}^\dagger\right\rangle, \ \hat{\xi}^\intercal = \left[\hat{a}_1, \ldots, \hat{a}_M, \hat{a}_1^\dagger, \ldots, \hat{a}_M^\dagger\right],$$

where $\hat{a}_i^\dagger$ and $\hat{a}_i$ are the bosonic creation and annihilation operators for some mode i, defined in the usual manner. σ is Hermitian and, for later convenience, we can write it in terms of the submatrices:

$$\sigma = \begin{pmatrix} V + I_M/2 & U \\ U^* & V^\intercal + I_M/2 \end{pmatrix}$$

such that V is Hermitian and U is symmetric; I_M is the $M \times M$ identity. As we shall see, $V \sim \langle \hat{a}_i^\dagger \hat{a}_j + h.c. \rangle$ contains the information pertaining to the state's classical properties and thermal correlations, and $U \sim \langle \hat{a}_i^\dagger \hat{a}_j^\dagger + h.c. \rangle$ encodes the entanglement and higher-order-correlation physics.

Note that it is more common in the quantum optics and information literature to see the quadrature covariance matrix, which use the real-valued canonical variables $\hat{x} \propto \hat{a} + \hat{a}^\dagger$ and $\hat{p} \propto \hat{a} - \hat{a}^\dagger$. The quadrature basis has the advantage of being a real-valued symplectic space. However, the $\hat{a}$-basis is a more natural convention for photon-number properties of the field.

Overall, the Gaussian states defined by σ have $2M^2$ free parameters (real numbers), plus an additional M local phase degree of freedom that may be ignored, as they have no effect on the photon number statistics.

Constructing and Decomposing the Covariance Matrix

Here we summarize some of the body of work pertaining to the matrix representations of multimode Gaussian systems. For more depth, see Refs. [190, 191, 192, 188, 193, 141, 189].

In this formalism, the space of Gaussian states is closed under Gaussian operations, some of which are represented as linear transformations. We call the transformation a Green's function if it represents the outcome of a process described by a linear differential equation. In a discrete basis of $\hat{a}$-operators, any

lossless Gaussian operation can be represented as:

$$\hat{a}_{\text{out}} = C\hat{a}_{\text{in}} + S\hat{a}_{\text{in}}^{\dagger}$$
$$\hat{\xi}_{\text{out}} = \begin{pmatrix} C & S \\ S^* & C^* \end{pmatrix} \hat{\xi}_{\text{in}} = G\hat{\xi}_{\text{in}}$$

or in a continuous basis parametrized by ω:

$$\hat{a}_{\text{out}} = \int C(\omega, \omega')\hat{a}_{\text{in}}(\omega') + S(\omega, \omega')\hat{a}_{\text{in}}(\omega')^{\dagger} d\omega'.$$

S may only be nonzero only if there is squeezing (if the Hamiltonian contains an $\hat{a}^{\dagger}\hat{a}^{\dagger} + c.c.$ term).

In our experiment, specifically, we are interested in the Green's function of the OPA process, a combination of parametric amplification and dispersion described by the differential equation:

$$\frac{d\hat{a}}{dz}(\Delta\omega) = iD(\Delta\omega)\hat{a} + iA(z, \Delta\omega) * \hat{a}^{\dagger}$$

where D represents the dispersion (phase matching, group velocity difference, and higher order dispersion), and the coupling term A represents the classical pump field, convolved with the quantum field. This equation is the same as you would find in classical nonlinear optics, but with the operator $\hat{a}$ replacing the classical field term. Classically, we could interpret this as the expectation value of the field operator and its evolution; quantum mechanically, we should think of this as the evolution of C, S that act on the operator. In our experiment, we expect S to be predominantly anti-diagonal due to energy conservation, and

C to depend on the phase-matching function.

Such a Green's function can be decomposed by the Bloch–Messiah decomposition, which informs us of how much squeezing there is and over what modes. To take advantage of standard computational methods, this is performed in the quadrature basis of $\hat{x}, \hat{p}$ (denote this Green's function by G'). This matrix decomposition returns canonically conjugate squeezed and anti-squeezed "supermodes." [1]

$$
\begin{aligned}
G' &= O_{\text{out}} \Sigma O_{\text{in}}^{\mathsf{T}} \\
\Sigma &= \tfrac{1}{2}\text{diag}(\{s_i\}_{i=1}^{M}, \{s_i^{-1}\}_{i=1}^{M}) = \tfrac{1}{2}\text{diag}(\{\mathrm{e}^{r_i}\}_{i=1}^{M}, \{\mathrm{e}^{-r_i}\}_{i=1}^{M})
\end{aligned}
$$

where s_i are the symplectic eigenvalues, related to the squeezing parameters r_i, and orthogonal matrices O contain the "input" and "output" symplectic eigenvectors or supermodes, which come in pairs (squeezed and anti-squeezed). The input supermodes are not important if the initial state is vacuum, as the squeezed vacuum only depends on the output supermodes. Intuitively, Bloch–Messiah may be thought of as a singular value decomposition that preserves commutation relations. These bases diagonalize the Green's function into single mode squeezing operations. Figure 6.2 illustrates the process. Each s_i represents an independent squeezed mode, and a source of photons with $\langle n \rangle = \sinh^2 r_i$ distributed over the mode $O_{\text{out},i}$, or from a quantum noise perspective, a $20 \log_{10} s_i$ dB noise reduction in the mode $O_{\text{out},i}$. A change of basis from $\hat{x}, \hat{p}$ back to $\hat{a}$ transforms the orthogonal matrices into unitary, and the diagonal matrix into a

[1] Note: this assumes the convention $\hat{x} = \hat{a}^\dagger + \hat{a}$ and $\hat{p} = i(\hat{a}^\dagger - \hat{a})$.

matrix where the quadrants are diagonal:

$$\sigma = \begin{pmatrix} \mathrm{diag}(\{\cosh r_i\}_{i=1}^{M}) & \mathrm{diag}(\{\sinh r_i\}_{i=1}^{M}) \\ \mathrm{diag}(\{\sinh r_i\}_{i=1}^{M}) & \mathrm{diag}(\{\cosh r_i\}_{i=1}^{M}) \end{pmatrix}.$$

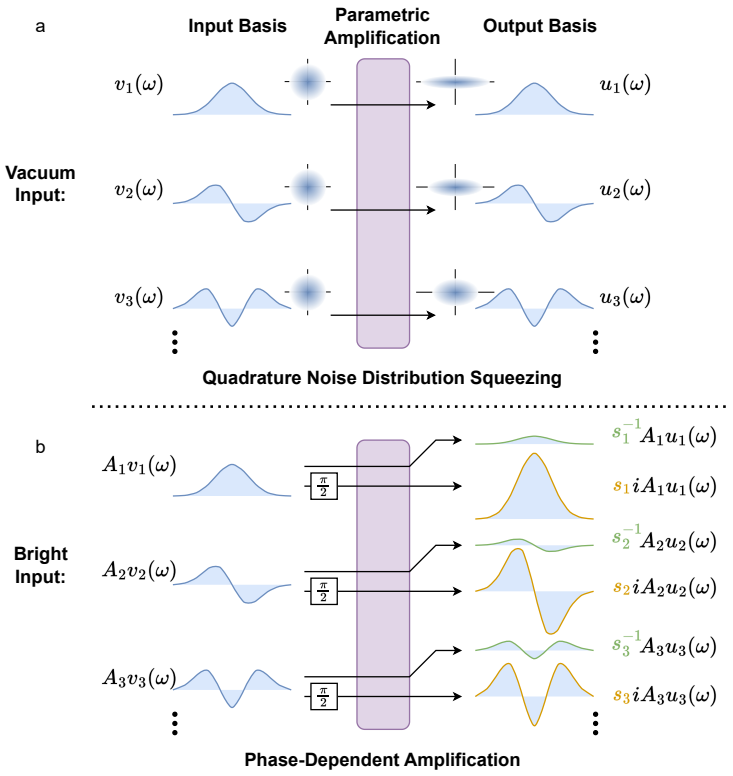


Figure 6.2: **Parametric amplification and the Bloch–Messiah decomposition.** The Bloch–Messiah decomposition reduces the parametric amplification to a phase-dependent amplification of an orthogonal set of modes. Each input mode is transformed to an output mode during this process: these modes are generally not the equal due to other effects in the OPA that may alter the signal (e.g. dispersion). **a.** In the case of vacuum input, squeezed vacuum is generated in each output mode independently. **b.** In the case of bright input, a signal may be decomposed into input modes, mapped to output modes, and each component is (de)amplified depending on the phase.

With vacuum input, a covariance matrix can be generated as follows:

$$\sigma_{ij} = \tfrac{1}{2}\{G_{ik}\hat{\xi}_k, G_{jk}\hat{\xi}_k\} \Rightarrow \sigma = \tfrac{1}{2}GG^\dagger.$$

Additional operations can be similarly applied to the covariance matrix:

$$\sigma \to G\sigma G^\dagger.$$

Note that vacuum $\sigma = I_{2M}/2$.

As previously mentioned, (going back to the $\hat{x}, \hat{p}$ basis) the input supermodes cancel out in the vacuum case:

$$\tfrac{1}{2}G'G'^\intercal = \tfrac{1}{2}O_{\text{out}}\Sigma O_{\text{in}}^\intercal O_{\text{in}}\Sigma O_{\text{out}}^\intercal = \tfrac{1}{2}O_{\text{out}}\Sigma^2 O_{\text{out}}^\intercal$$

and we see that the Bloch–Messiah decomposition is also valid on the the covariance matrix, although one must account for squared diagonal matrix (however, we note that the Williamson decomposition must be used when the state is not pure: in the case of loss, thermal noise, or if the state has been partially traced out).

We can observe that:

$$\begin{aligned}\sigma = \tfrac{1}{2}GG^\dagger &= \frac{1}{2}\begin{pmatrix} SS^\dagger + CC^\dagger & CS^\intercal + SC^\intercal \\ C^*S^\dagger + S^*C^\dagger & S^*S^\intercal + C^*C^\intercal \end{pmatrix} \\ &= \begin{pmatrix} SS^\dagger + I/2 & (CS^\intercal + SC^\intercal)/2 \\ (C^*S^\dagger + S^*C^\dagger)/2 & S^*S^\intercal + I/2 \end{pmatrix} = \begin{pmatrix} V + I/2 & U \\ U^* & V^\intercal + I/2 \end{pmatrix}.\end{aligned}$$

Note that for the U quadrants to be nonzero, a squeezing operation must be involved through a nonzero S. No unitary C applied to any thermal state can populate U. On the other hand, the squeezing contributes to "thermal" components V, as it adds photons to the field, thus increasing the photon-number variance.

The formalism introduced so far – where all the operations are unitary or symplectic and preserve commutation relations – cannot account for losses or inefficiencies that must be considered during the frequency conversion and detection steps of our experiment. In both cases, the state occupies unobserved modes, that are traced out. Tracing out modes in a covariance matrix is simply equivalent to removing the corresponding rows and columns, in other words, taking a principal submatrix that omits the traced out modes. For example, this is used to derive the action of a lump loss or noise on a covariance matrix (modeled as passing through a fictitious beamsplitter and tracing out the second port):

$$\sigma \rightarrow \sqrt{\eta\eta^{\intercal}} \circ \sigma + (1-\eta) \circ \nu \circ I$$

where η is the transmission, $\nu = \bar{n} + 1/2$ represents any thermal noise added ($1/2$ for vacuum). Both are vectors in the general case. The $\circ$ operator is the element-wise or Hadamard product.

We now use these results to describe the physics relevant to our experiment: we convert the infrared squeezed light to visible, but do not detect the remaining infrared light. For notational simplicity let $\hat{\xi} = [\hat{a}_{\text{vis}}, \hat{a}^{\dagger}_{\text{vis}}, \hat{a}_{\text{ir}}, \hat{a}^{\dagger}_{\text{ir}}]$. The initial state is:

$$\sigma_{\text{tot}}(z=0) = \begin{pmatrix} \sigma_{\text{vis}} & \sigma_{\text{vis,ir}} \\ \sigma^{\dagger}_{\text{vis,ir}} & \sigma_{\text{ir}} \end{pmatrix} = \begin{pmatrix} I_{2M}/2 & 0 \\ 0 & \sigma_{\text{ir}}(0) \end{pmatrix}.$$

The AFC (over crystal length L) acts as a unitary on the entire state:

$$\sigma_{\text{tot}}(z = L) = \begin{pmatrix} G_{\text{vis,vis}} & G_{\text{vis,ir}} \\ G_{\text{ir,vis}} & G_{\text{ir,ir}} \end{pmatrix} \begin{pmatrix} I_{2M}/2 & 0 \\ 0 & \sigma_{\text{ir}} \end{pmatrix} \begin{pmatrix} G_{\text{vis,vis}} & G_{\text{vis,ir}} \\ G_{\text{ir,vis}} & G_{\text{ir,ir}} \end{pmatrix}^{\dagger}.$$

Expanding and tracing out the infrared modes we obtain the covariance matrix of the observed visible modes:

$$\sigma_{\text{vis}}(L) = G_{\text{vis,ir}}\sigma_{\text{ir}}(0)G^{\dagger}_{\text{vis,ir}} + G_{\text{vis,vis}}G^{\dagger}_{\text{vis,vis}}/2.$$

As the conversion tends to unity, $G_{\text{vis,ir}}$ becomes unitary and $G_{\text{vis,vis}}$ vanishes.

The sum frequency generation equations that yield the Green's function in this case are:

$$\frac{d\hat{a}_{\text{vis}}}{dz}(\Delta\omega) = iD_{\text{vis}}(z, \Delta\omega)\hat{a}_{\text{vis}} + iA(z, \Delta\omega) * \hat{a}_{\text{ir}}$$
$$\frac{d\hat{a}_{\text{ir}}}{dz}(\Delta\omega) = iD_{\text{ir}}(z, \Delta\omega)\hat{a}_{\text{ir}} + iA^{*}(z, \Delta\omega) * \hat{a}_{\text{vis}}$$

where, again, A is the pump field and D is the dispersion, which is notably a function of z, due to aperiodic poling.

Photon Number Properties of Zero-Mean Gaussian States

For convenience, in the context of photon number statistics, we can make the covariance matrix symmetric and complex-valued, by defining the transformation

X, as first introduced in [144]:

$$\sigma' = \sigma - I_{2M}/2, \quad X\sigma' = \begin{pmatrix} U & V^\intercal \\ V & U^* \end{pmatrix},$$
$$X = \begin{pmatrix} 0 & I_M \\ I_M & 0 \end{pmatrix}.$$

σ' defined in this manner is useful, because we can use it to find any of the following photon number expectation values:

$$\left\langle \prod_i \hat{n}_i^{m_i} \right\rangle = \left\langle \prod_i \left(\hat{a}_i^\dagger \hat{a}_i\right)^{m_i} \right\rangle = \mathrm{Haf}(X\sigma'_m), \; m \in \{0,1\}^M.$$

using the conventional definition of the Hafnian (Haf). σ'_m indicates the principal submatrix with indices given by m. Suffice to say, the Hafnian function is central to the (Gaussian or perturbative) multimode physics of bosons, since it can be thought of as an implementation of Wick's theorem for Gaussian integrals; for zero-mean Gaussian variables x_{i_j}:

$$\langle x_{i_1} \dots x_{i_{2m}} \rangle = \sum_{\mathcal{P}} \langle x_{k_1} x_{k_2} \rangle \dots \langle x_{k_{2m-1}} x_{k_{2m}} \rangle$$

where the sum is over all pairings $\mathcal{P}$ – all possible ways to group the i-indices into m pairs of k-indices – hence the Hafnian, a function defined for this very purpose [194]. And of course the operators $\hat{a}_i$ are jointly Gaussian (in their quasi-probability distributions).

The general case with $m \in \mathbb{N}^M$ is more complicated, as it requires more careful consideration of operator ordering. Indeed, we were being hasty: $\hat{a}$ is not a

random variable, it is an operator. However, Wick's theorem tells us that this was allowed as long as the state and operator have compatible representations. For example, one way to "treat our operators as though random variables" is by converting our expression to Weyl ordering (denoted by $:\ldots:_W$). This requires a prescribed order of $\hat{a}$'s and $\hat{a}^\dagger$'s, and once we have this expression, we may move the operators within without incurring a commutation relation, e.g. $:\hat{a}^\dagger\hat{a}:_W = :\hat{a}\hat{a}^\dagger:_W$. The correct procedure is to expand out the operator expression into Weyl-ordered expressions, and replace each one with the corresponding Hafnian, i.e.:

$$\begin{aligned}
\langle \hat{n}_1^{m_1}\hat{n}_2^{m_2}\ldots\rangle &= \langle (\hat{a}_1^\dagger\hat{a}_1)^{m_1}(\hat{a}_2^\dagger\hat{a}_2)^{m_2}\ldots\rangle \\
&= \sum_{\text{orderings}} c_{\text{order}}\langle :\hat{a}_1^{\dagger m_{\text{order1}}}\hat{a}_1^{m_{\text{order2}}}\ldots:_W\rangle \\
&= \sum_{\text{orderings}} c_{\text{order}}\left[\sum_{\mathcal{P}(\text{order})}\prod_{(k_1,k_2)}\langle :\hat{a}_{k_1}\hat{a}_{k_2}:_W\rangle\right] \\
&= \sum_{\text{orderings}} c_{\text{order}}\text{Haf}(X\sigma_{\bar{m}_{\text{order}}}).
\end{aligned}$$

It is well known how to relate normal (operator expressions where all the $\hat{a}^\dagger$ precede the $\hat{a}$, denoted by $:\ldots:$) and Weyl forms [195, 189], in our case:

$$:\hat{a}^{\dagger m}\hat{a}^m: = \sum_{k=0}^{m} k!\binom{m}{k}^2\left(-\frac{1}{2}\right)^k :\hat{a}^{\dagger m-k}\hat{a}^{m-k}:_W$$

and all modes are treated separately. Additionally, the normal form of $(\hat{a}^\dagger\hat{a})^m$ can be obtained by repeatedly applying commutation relations. This can be

generalized to a sum represented by Stirling numbers S [196]:

$$(\hat{a}^\dagger \hat{a})^m = \sum_{i=1}^{m} S(m,i) \hat{a}^{\dagger i} \hat{a}^i.$$

In the case of $m \in \{0,1\}^M$, as above, $\hat{a}^\dagger \hat{a} = :\hat{a}^\dagger \hat{a}:_W - 1/2$, and we can subtract the constant directly from the covariance matrix as we did for σ' (σ is conventionally defined as above, with the anticommutator, such that it is conveniently in Weyl form).[2] Similarly, for the second order case,

$$\langle \hat{n}^2 \rangle = \langle (\hat{a}^\dagger \hat{a})^2 \rangle = \langle :\hat{a}^{\dagger 2} \hat{a}^2 + \hat{a}^\dagger \hat{a}: \rangle = \langle :\hat{a}^{\dagger 2} \hat{a}^2 - \hat{a}^\dagger \hat{a}:_W \rangle \Rightarrow \mathrm{Haf}(X\sigma^{(2)}) - \mathrm{Haf}(X\sigma).$$

In this example we use $\sigma^{(2)}$ to denote a larger covariance matrix where we repeat the rows and columns of σ for this mode.

These equations involving Hafnians will look familiar to the reader familiar with the recent Gaussian Boson Sampling literature. Indeed, for multimode Gaussian states, calculating the probabilities is largely the dual of calculating the expectation values. The photon number distribution is essentially a generalized Bose–Einstein distribution that incorporates modes, interference and entanglement. To quote the result proven in [143, 144], the probability of a given measurement, $n \in \mathbb{N}^M$, has the form:

$$P(n) = \frac{1}{n! |\sigma + I_{2M}/2|^{1/2}} \mathrm{Haf}(XA_n)$$
$$A = I_{2M} - (\sigma + I_{2M}/2)^{-1}, \quad n! = \prod_i n_i!$$

[2] Note, however, that because in this case $\hat{n}^1 = \hat{a}^\dagger \hat{a}$ is already normal ordered, we end up implicitly converting σ to normal ordering by subtracting $I/2$ (i.e. $\sigma'_{ii} = \langle :\hat{a}_i^\dagger \hat{a}_i: \rangle = \langle \hat{a}_i^\dagger \hat{a}_i \rangle$ from $\sigma_{ii} = \langle :\hat{a}_i^\dagger \hat{a}_i:_W \rangle = \langle \{\hat{a}_i^\dagger, \hat{a}_i\} \rangle$).

where the Hafnian's argument refers to the principal submatrix with indices given by n. To gain some intuition, one may notice the Bose–Einstein resemblance via $\sigma_{ii} = \langle \hat{a}_i^\dagger \hat{a}_i + h.c. \rangle / 2 = \langle n_i \rangle + 1/2$, so $(\sigma + I/2)_{ii} = \langle n_i \rangle + 1$, thus in some limiting cases:

$$|\sigma + I/2|^{-1/2} \to \frac{1}{\langle n \rangle + 1}, \quad I - (\sigma + I/2)^{-1} \to \frac{\langle n \rangle}{\langle n \rangle + 1}$$

More precisely, the Bose–Einstein distribution, which is simply geometric, can be thought of as stemming from the recurrence relation:

$$P_{\mathrm{BE}}(n) = P_{\mathrm{BE}}(n-1)a = P_{\mathrm{BE}}(0)a^n = \frac{1}{\langle n \rangle + 1} \left(\frac{\langle n \rangle}{\langle n \rangle + 1} \right)^n$$

while this general version follows the similar, yet more complex, Hafnian (Wick's theorem) recursive property:

$$\forall i \text{ s.t. } n_i > 0, P(n_1, \ldots, n_M) = \frac{P(0, \ldots, 0)}{n!} \sum_{\substack{j=1 \\ j \neq i}}^{2N} a_{ij} \mathrm{Haf}(X A_{n-\{i,j\}}), \quad N = \sum_i n_i$$

where the notation $A_{n-\{i,j\}}$ denotes the subtraction of rows and columns i, j from the matrix A_n (note e.g. with $j = i + N$, $\mathrm{Haf}(A_{n-\{ij\}}) \propto P(n_1, \ldots, n_i - 1, \ldots, n_M)$, $a_{ij} = P(0, \ldots, n_i = 1, \ldots, 0)$).

From the quantum computing and complexity theory perspective, the Hafnian is of interest as it belongs to #P, a class of classically intractable functions [197, 144, 145].

A brief sketch of a derivation of the probabilities is as follows; it is perhaps a bit more cumbersome than using phase space formalism (which abstracts away

the use of operators and orders) but it is hopefully more transparent. We use the normal form, as this allows us to use a convenient representation of the photon-number Fock state projection operator: the vacuum projection is known to be

$$|0\rangle\langle 0| = :\exp(-\hat{a}^\dagger \hat{a}):$$

(see, for example, [198] Eqns. 17–19 for a short proof), and it follows that [189]:

$$|n\rangle\langle n| = |n_1, \ldots, n_m\rangle\langle n_1, \ldots, n_m| = \prod_i (\hat{a}_i^\dagger)^{n_i} |0\rangle\langle 0| \prod_i \hat{a}_i^{n_i}/n_i! = :\prod_{i=1}^{M} \exp(-\hat{a}_i^\dagger \hat{a}_i) \frac{\left(\hat{a}_i^\dagger \hat{a}_i\right)^{n_i}}{n_i!}:$$

$$P(n) = \langle |n\rangle\langle n| \rangle = \frac{1}{n!} \left\langle :\prod_{i=1}^{M} \exp(-\hat{a}_i^\dagger \hat{a}_i) \left(\hat{a}_i^\dagger \hat{a}_i\right)^{n_i}: \right\rangle = \left\langle :\prod_{i=1}^{M} \frac{\mathrm{e}^{-\hat{n}_i} \hat{n}_i^{n_i}}{n_i!}: \right\rangle.$$

Since the operator is normal-ordered, we must also use the normal-ordered covariance matrix, which is σ' as defined above (since, e.g. for the first matrix quadrant, $\sigma'_{ij} = \langle :\hat{a}_i^\dagger \hat{a}_j: \rangle = \langle \hat{a}_i^\dagger \hat{a}_j \rangle = \langle \{\hat{a}_i^\dagger, \hat{a}_j\} \rangle - \delta_{ij} = \langle :\hat{a}_i^\dagger \hat{a}_j:_W \rangle - \delta_{ij} = \sigma_{ij} - \delta_{ij}$). Lastly, notice that we can set

$$|0\rangle\langle 0| = :\exp\left(-\tfrac{1}{2}\hat{\xi}^\dagger \hat{\xi}\right):$$

so that the notation is compatible with that of the state's covariance matrix.

Hence, the problem is reduced once more to calculus; multivariate Gaussian integrals. We first find $P(0)$, and the rest of the probabilities follow from Wick's theorem, with a modified covariance matrix due to the above $|0\rangle\langle 0|$ Gaussian. The former can be solved with the usual tricks. Let $Z \sim \mathcal{N}(0, I)$ be a vector of standard normal random variables. By affine transformation, we can transform it to a vector of arbitrary jointly Gaussian variables: $\sigma'^{1/2} Z + \mu$. Since we are only interested in the case of zero displacement, let the means $\mu = 0$. We thus simplify our calculation by computing the expectation over the multivariate standard

normal distribution:

$$\left\langle \exp\left(-\tfrac{1}{2}(\sigma'^{1/2}Z)^\dagger \sigma'^{1/2}Z\right)\right\rangle = (2\pi)^{-M}\int_{\mathbb{R}^{2M}} dZ \exp\left(-\tfrac{1}{2}Z^\dagger(\sigma'+I)Z\right) = |\sigma'+I|^{-1/2}.$$

This is the zero-photon probability. We note that $\sigma' + I = \sigma + I/2$ corresponds to the anti-normal covariance matrix (all the $\hat{a}$ precede the $\hat{a}^\dagger$).

Finally, we can include monomial terms in the integral to calculate other probabilities, using Wick's theorem once more, after solving for the effective covariance matrix, A – since the distribution and operator Gaussians combine:

$$\exp\left(-\tfrac{1}{2}Z^\dagger Z - \tfrac{1}{2}Z^\dagger \sigma'^{-1}Z\right) = \exp\left(-\tfrac{1}{2}Z^\dagger A^{-1}Z\right).$$

A can be found by invoking the matrix inversion lemma:

$$\begin{aligned} A^{-1} &= \sigma'^{-1} + I \\ A &= (\sigma'^{-1} + I)^{-1} \\ &= I - \sigma'^{-1}(\sigma^{-1} + I)^{-1} \\ &= I - (\sigma' + I)^{-1}, \end{aligned}$$

as per convention. We have not been completely rigorous, however, since σ' may not be invertible (if, in a diagonal basis, sigma has some $\langle :\hat{a}_i^\dagger \hat{a}_i: \rangle = 0$, [3] in which case the Gaussian tends towards a delta function); hence, besides for the last line, the above equalities may not be strictly valid. Nonetheless, the moments exist, which is what we need for the moment expansion in Wick's

[3] The field can never be a pure vacuum at finite temperature, although for all intents and purposes it is at optical frequencies, since the photon energy $\hbar\omega \gg k_B T$ (the Boltzmann constant temperature product) at room temperature.

theorem.

Photon Number Expectation Values

We can solve for some lower order photon number moments in terms of σ's elements. To eliminate local phase degrees of freedom, we set all diagonal U_{ii} terms to be real and positive. Here, wherever relevant, we have $i < j < k$ (switching order requires taking complex conjugates of some terms and can therefore introduce inconsistencies).

$$
\begin{aligned}
\langle n_i \rangle &= \mathrm{Haf}(X\sigma'_i) = V_{ii} \\
\langle n_i n_j \rangle &= \mathrm{Haf}(X\sigma'_{ij}) = |U_{ij}|^2 + |V_{ij}|^2 + \langle n_i \rangle \langle n_j \rangle \\
\langle n_i^2 \rangle &= \mathrm{Haf}(X\sigma_{ii}) - \mathrm{Haf}(X\sigma_i) = |U_{ii}|^2 + 2\langle n_i \rangle^2 + \langle n_i \rangle \\
\langle n_i n_j n_k \rangle &= \mathrm{Haf}(X\sigma'_{ijk}) \\
&= \langle n_i \rangle \langle n_j n_k \rangle + \langle n_j \rangle \langle n_i n_k \rangle + \langle n_k \rangle \langle n_i n_j \rangle - 2\langle n_i \rangle \langle n_j \rangle \langle n_k \rangle \\
&\quad + 2\mathcal{R}\left(U^*_{ij}(V_{ik}U_{jk} + V_{jk}U_{ik}) + V_{ij}(U^*_{ik}U_{jk} + V^*_{ik}V_{jk}) \right) \\
\langle n_i^2 n_j \rangle &= \mathrm{Haf}(X(\sigma_{iij} - \tfrac{1}{2}\delta_{jj}) - \mathrm{Haf}(X(\sigma_{ij} - \tfrac{1}{2}\delta_{jj})) \\
&= \langle n_i n_j \rangle (4\langle n_i \rangle + 1) + \langle n_j \rangle (|U_{ii}|^2 - 2\langle n_i \rangle^2) + 2U_{ii}(U^*_{ij}V_{ij} + U_{ij}V^*_{ij}) \\
\langle n_i n_j^2 \rangle &= \mathrm{Haf}(X(\sigma_{ijj} - \tfrac{1}{2}\delta_{ii}) - \mathrm{Haf}(X(\sigma_{ij} - \tfrac{1}{2}\delta_{ii})) \\
&= \langle n_i n_j \rangle (4\langle n_j \rangle + 1) + \langle n_i \rangle (|U_{jj}|^2 - 2\langle n_j \rangle^2) + 2U_{jj}(U_{ij}V_{ij} + U^*_{ij}V^*_{ij}) \\
\langle n_i^2 n_j^2 \rangle &= \mathrm{Haf}(X\sigma_{iijj}) + \mathrm{Haf}(X\sigma_{ij}) - \mathrm{Haf}(X\sigma_{iij}) - \mathrm{Haf}(X\sigma_{ijj}) \\
&= \langle n_i^2 n_j \rangle (4\langle n_j \rangle + 1) + \langle n_i n_j^2 \rangle (4\langle n_i \rangle + 1) - \langle n_i n_j \rangle (4\langle n_i \rangle + 1)(4\langle n_j \rangle + 1) \\
&\quad + 4(\langle n_i n_j \rangle - \langle n_i \rangle \langle n_j \rangle)^2 + (2\langle n_i \rangle^2 - |U_{ii}|^2)(2\langle n_j \rangle^2 - |U_{jj}|^2) \\
&\quad + 2U_{ii}U_{jj}(V_{ij}^2 + U_{ij}^2 + V_{ij}^{*\,2} + U_{ij}^{*\,2}) + 8|U_{ij}|^2|V_{ij}|^2.
\end{aligned}
$$

A few important observations. The equations for the statistics can be separated into a trivial component (composed of lower order statistics) and an interference term, due to complex-valued U and V terms. If the U terms are zero, there are no nontrivial higher order correlations: everything can be described by second- or first-order statistics. Hence why thermal states do not have interesting higher order correlations.

Relatedly, where $U = 0$, there are a vast number of states that have the same photon number distribution, as there is no information beyond photon number mean and covariance. Measurements carry no information about the underlying supermodes.

The value of $U_{ii} \in [0, \langle n_i \rangle (\langle n_i \rangle + 1)]$, and the two extremes correspond to two-mode and single-mode squeezing respectively. States that have intermediate values can be thought of as a multimode generalization of these two concepts, where the value of U_{ii} represents the locality of the entanglement. When $U_{ii} = 0$, certain higher order correlations also lose interference terms. Therefore, measurements of two-mode squeezed vacuum also lack certain information about the underlying state.

Finally, for a state with sufficient information encoded in U, the inverse problem of retrieving the state and modes from observations is possible, within some limitations. This is within the realm of possibility and future experiments should seek to achieve this.

6.1.3 Coincidence Detection

In this section we explain how coincidence detection may be used as a tool to distinguish between different pulsed photonic states, as well as measure the purity of the state.

Derivation

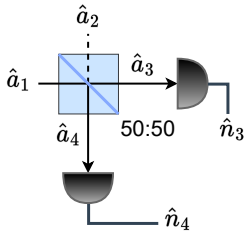


Figure 6.3: **Toy model for the coincidence detection experiment.**

Consider a beam of squeezed light incident on a balanced beamsplitter, which is followed by two detectors on each side of the output (Figure 6.3). The first arm of the beamsplitter, with operator $\hat{a}_1$, has some $\langle\hat{n}\rangle$ and $(\hat{n}) = 2\langle\hat{n}\rangle(\langle\hat{n}\rangle+1)$, in the lossless case ($\eta = 1$). The second input port, with operator $\hat{a}_2$ has vacuum input.

The third and fourth ports, defined by operators $\hat{a}_3, \hat{a}_4$, have $\langle\hat{n}_3\rangle = \langle\hat{n}_4\rangle =$

$\langle\hat{n}\rangle/2$. Evaluating their statistics:

$$\begin{aligned}
\langle\hat{n}_3\hat{n}_4\rangle &= \left\langle \hat{a}_3^\dagger\hat{a}_3\hat{a}_4^\dagger\hat{a}_4 \right\rangle \\
&= \frac{1}{4}\left\langle (\hat{a}_1+\hat{a}_2)^\dagger(\hat{a}_1+\hat{a}_2)(\hat{a}_1-\hat{a}_2)^\dagger(\hat{a}_1-\hat{a}_2)\right\rangle \\
&= \frac{1}{4}\left\langle \hat{n}_1^2 - \hat{a}_2\hat{a}_2^\dagger\hat{n}_1 \right\rangle \\
&\Rightarrow \frac{1}{4}\left(\left\langle \hat{n}_1^2\right\rangle - \langle\hat{n}_1\rangle\right), \eta = 1
\end{aligned}$$

We can now compute the covariance of the number measurements on each detector.

$$\begin{aligned}
(\hat{n}_3,\hat{n}_4) &= \langle\hat{n}_3\hat{n}_4\rangle - \langle\hat{n}_3\rangle\langle\hat{n}_4\rangle \\
&= \frac{1}{4}\left(\left\langle \hat{n}_1^2\right\rangle - \langle\hat{n}_1\rangle\right) - \frac{1}{4}\langle\hat{n}_1\rangle^2 \\
&= \frac{1}{4}\left((\hat{n}_1) - \langle\hat{n}_1\rangle\right) \\
&\Rightarrow \frac{1}{4}\langle\hat{n}\rangle(2\langle\hat{n}\rangle+1\rangle),\ \eta = 1
\end{aligned}$$

Note that for coherent states, with $(\hat{n}) = \langle\hat{n}\rangle$, and for thermal states, with $(\hat{n}) = \langle\hat{n}\rangle(\langle\hat{n}\rangle + 1)$, the covariances are 0 and $\langle\hat{n}\rangle^2/4$, respectively. Similarly, Fock states with $(\hat{n}) = 0$, have negative covariance; this is the well-known anti-bunching behavior. These different scaling behaviors allow us to experimentally distinguish these different photonic states.

To account for loss, we introduce a fictitious unbalanced beamsplitter oper-

ation such that $\langle n_1' \rangle = \eta \langle n_1 \rangle$:

$$\begin{aligned}(\hat{n}_1') &= \left\langle \left(\sqrt{\eta}\hat{a}_1 + \sqrt{1-\eta}\hat{a}_0 \right)^\dagger \left(\sqrt{\eta}\hat{a}_1 + \sqrt{1-\eta}\hat{a}_0 \right) \left(\sqrt{\eta}\hat{a}_1 + \sqrt{1-\eta}\hat{a}_0 \right)^\dagger \left(\sqrt{\eta}\hat{a}_1 + \sqrt{1-\eta}\hat{a}_0 \right) \right\rangle - \eta^2 \langle \hat{n}_1 \rangle^2 \\ &= \eta^2 \left\langle \hat{n}_1^2 \right\rangle + \eta(1-\eta)\langle \hat{n}_1 \rangle - \eta^2 \langle n_1 \rangle^2 \\ &= \eta^2 (\hat{n}_1) + \eta(1-\eta)\langle \hat{n}_1 \rangle\end{aligned}$$

So the covariance will simply change by a factor η^2:

$$\begin{aligned}(\hat{n}_3, \hat{n}_4) &= \frac{1}{4}\left((\hat{n}_1') - \langle \hat{n}_1' \rangle\right) \\ &= \frac{\eta^2}{4}\left((\hat{n}_1) - \langle \hat{n}_1 \rangle\right) \\ &\Rightarrow \frac{\eta^2}{4}\left(2\langle \hat{n}_1 \rangle^2 + \langle \hat{n}_1 \rangle\right)\end{aligned}$$

If we consider the case of asymmetric loss, it can be shown that the transmission efficiency can be replaced by the individual efficiencies of the left and right arms, $\eta^2 \rightarrow \eta_L \eta_R$.

Finally, in the multimode case, the total covariance is simply the sum of the covariances of all the individual modes (as a consequence of mode independence/commutation). Therefore, a lossy multimode squeezed state has:

$$\begin{aligned}(\hat{n}_3, \hat{n}_4) &= \sum_i \frac{\eta_{Li}\eta_{Ri}}{4}\left((\hat{n}_i) - \langle \hat{n}_i \rangle\right) \\ &\Rightarrow \sum_i \frac{\eta_{Li}\eta_{Ri}}{4}\left(2\langle \hat{n}_i \rangle^2 + \langle \hat{n}_i \rangle\right)\end{aligned}$$

In the case where all modes have $\langle \hat{n}_i \rangle \ll 1$, the covariance is a linear function

of the photon number

$$(\hat{n}_3, \hat{n}_4) \approx \sum_i \frac{\eta_{Li}\eta_{Ri}}{4}\langle \hat{n}_i \rangle$$

Notably, the slope is determined by the overall transmission efficiency, $\eta/2$ (assuming a constant transmission), since the average number of photons per detector is

$$\langle N \rangle = \sum_i \frac{\eta_i}{2}\langle \hat{n}_i \rangle$$

Therefore, in addition to being a test for squeezing, this experiment may also serve as a measurement of loss.

To measure loss at slightly larger photon numbers, a quadratic fit may be used:

$$(\hat{n}_3, \hat{n}_4) \approx \frac{\eta}{2}\langle N \rangle + \frac{\eta}{2} C \langle N \rangle^2, \quad C \approx \sum_i \left(\frac{\eta_i \langle n_i \rangle}{\langle N \rangle} \right)^2$$

so long as $\langle n_i \rangle / \langle N \rangle \approx$ const. In other words, the photon numbers in the state must scale linearly. This requires that the squeezed modes remain in the region where $\langle n \rangle = \sinh^2 r_i \approx r_i^2$. Once again, the transmission is given by the linear coefficient.

Threshold Detection

Finally, we must consider the case of threshold detectors, where the detector only provides "clicks," if it registers any number of photons. This reduces the two observables to Bernoulli variables. While the total number of photons

$\langle N \rangle \ll 1$, the physics remains the same. However, the covariance of Bernoulli variables is bound by a parabola, therefore the photon covariance will behave as such:

$$\begin{aligned}(n_3, n_4) &= \langle n_3 n_4 \rangle - \langle n_3 \rangle \langle n_4 \rangle \\ &= P(n_3 = 1 | n_4 = 1) P(n_4 = 1) \\ &\quad - \left(P(n_3 = 1 | n_4 = 1) P(n_4 = 1) + P(n_3 = 1 | n_4 = 0) P(n_4 = 0)\right) \langle n_4 \rangle \\ &= \left(P(n_3 = 1 | n_4 = 1) + P(n_3 = 1 | n_4 = 0)\right) (1 - \langle n_4 \rangle) \langle n_4 \rangle\end{aligned}$$

where we expect the probabilities to be constant in the many mode, $\langle \hat{n}_i \rangle \ll 1$, scenario, where coincidences are predominantly due to biphotons.

Predicting Coincidence Outcomes with Wavelength-Dependent Detector Quantum Efficiency

The above treatment of coincidence detection assumes a uniform quantum efficiency. However, in our infrared-wavelength experiment, we have that $\eta_{Li}, \eta_{Ri} \to \eta_{Li}(\lambda), \eta_{Ri}(\lambda)$. This thermalizes the state further than uniform loss. Because the QE cutoff is 1700 nm, any photon above this wavelength will never be detected, it is traced out and its sister photon therefore effectively becomes a thermal photon. In addition, the spectrum measured in experiment is far broader than in simulation. Therefore, while it is straightforward to derive the covariance expected for a given QE for the simulated state, the simulation is not representative of experiment in this case.

To account for this, we derive a simple model to estimate the photon number vs covariance scaling we expect in the experiment, using the spectrum mea-

sured experimentally. In addition, we use the specified detector quantum efficiency.

Assume that the number of modes $M \gg \langle N \rangle$, the total number of photons, and therefore the probability of any given mode yielding photon pairs, $P_m \ll 1$, where m indexes the mode. We truncate the wavefunction of each mode in the photon number basis, as per the biphoton approximation:

$$|\psi\rangle_m \approx \sqrt{1 - P_m}|0\rangle_m + \sqrt{P_m}|2\rangle_m + O(P_m)$$

Furthermore assume that the probability of producing more than two photons per event is negligible. Then we can define the probability $\mathcal{P}_m$ that detected photons arise from any given mode:

$$\langle N \rangle = 2 \sum_m P_m$$
$$\sum_m \mathcal{P}_m = 1 \Rightarrow \mathcal{P}_m = \frac{2}{\langle N \rangle} P_M$$

The covariance is a sum over all covariances, which depend on the coincidence of two photons on the two different detectors:

$$\begin{aligned}(\hat{n}_3, \hat{n}_4) = \sum_m {}_m(\hat{n}_3, \hat{n}_4) &\approx \sum_m \frac{1}{2} P_m \eta_L \eta_R \mathrm{QE}(\omega_{m,1}) \mathrm{QE}(\omega_{m,2}) + O(P_m^2) \\ &= \sum_m \langle N \rangle \mathcal{P}_m \eta_L \eta_R \mathrm{QE}(\omega_{m,1}) \mathrm{QE}(\omega_{m,2}) + O(P_m^2)\end{aligned}$$

By switching to the continuous frequency basis and parametrizing by $\Delta\omega$

about the central wavelength ω_0, we can obtain the expression:

$$\begin{aligned}(\hat{n}_3, \hat{n}_4) &\approx \langle N\rangle\eta^2 \int_0^{\Delta\omega_{\max}} d\Delta\omega \mathcal{P}(\Delta\omega)\mathrm{QE}(\omega_0 - \Delta\omega)\mathrm{QE}(\omega_0 + \Delta\omega) \\ &= \langle N\rangle\eta^2 \int_{\lambda_0}^{\lambda_{\min}} \frac{d\lambda}{\lambda^2}\left[\mathcal{P}(\lambda)\mathrm{QE}(\lambda)\mathrm{QE}\left(\frac{2}{\lambda_0} - \frac{1}{\lambda}\right)\right] \Big/ \int_{\lambda_0}^{\lambda_{\min}} \frac{d\lambda}{\lambda^2}\mathcal{P}(\lambda)\end{aligned}$$

where $\mathcal{P}(\lambda)$ is now the spectral density (corresponding to Figure **??**c in the experiment). As expected, the covariance is linear in $\langle N\rangle$.

For simplicity, it is assumed above that the biphotons are perfectly correlated in frequency, which for a finite bandwidth pump and finite number of supermodes is not physically correct, but a reasonable approximation for a narrow phase-matching bandwidth (with respect to $d\mathrm{QE}/d\omega$).

6.2 Highly Multimode Single-Photon Spectrometer

The infrared wavelength spectrometer is based on a diffraction grating on motorized rotation stage (Thorlabs GR25-0616; K10CR1). The first order reflection couples into SMF-28 single mode fiber (Thorlabs F260APC-1550), which monochromates the input. The fiber coupling efficiency is approximately 35%. For single photon detection, this coupled into a InGaAs SPAD (IDQuantique ID Qube NIR Gated), set to the nominal 15% QE. The wavelength-angle correspondence is calibrated using tunable lasers (JDSU mTLG-C1C1L1) between 1527 nm and 1609 nm and fitting to the grating equation, $m\lambda = d(\sin(\theta_{i0} + \Delta) - \sin(\theta_{ir} - \Delta) = 2d\cos[(\theta_{i0} + \theta_{r0})/2]\sin[(\theta_{i0} - \theta_{r0})/2 + \Delta]$, where d is the grating constant, θ_{i0}, θ_{r0} are some reference incidence and reflection angles, and Δ is the rotation angle of the grating. The extrapolation to uncalibrated wavelengths is deemed correct

as the spectrum stops sharply at 1150 nm, which matches the long pass filter cutoff (Thorlabs FELH1150). However, the coupling efficiency as a function of wavelength uncalibrated.

The visible light spectrometer is based on a diffraction grating (Ibsen PCG-1908/675-972) imaged by an objective lens (Olympus UPLFLN4x) onto the NüVü HNü 512 IS EMCCD camera. An EM gain of 3000 is used on the camera.

The wavelength-pixel correspondence is calibrated by monochromating supercontinuum (NKT Origami, Thorlabs HN1550) with the infrared monochromator, and converting this light through AFC. The AFC pump is amplitude-modulated to a narrow bandwidth, thus also effectively monochromated. The monochromated supercontinuum wavelength-to-angle is calibrated with an optical spectrum analyzer (Ando AQ6317B).

The SPAD and camera are triggered by the Amplitude Satsuma laser, with an appropriate time delay (IDQuantique ID900).

6.2.1 Spectrometer Design

In order to verify the correlation structure between frequency modes generated by our quantum light source, we design a spectrometer that has a uniformly high spectral resolution and low loss around λ_0 = 620 nm, the central wavelength. Here we outline the design considerations to achieve the desired spectral resolution.

The number of frequency modes we can resolve is ultimately limited by the

number of pixels in each row of the EMCCD camera. We choose an EMCCD camera with a sensor size of 512 by 512, instead of a larger one (e.g., 1024 by 1024), because the better signal-to-noise ratio that would likely preserve more information on our light source (see below for considerations in the choice of camera). The spectrometer is designed for a 60 nm bandwidth (a spectrum between 590 nm and 650 nm). The ideal spectral bin width in each EMCCD pixel is thus $\Delta\lambda = 60/512 \approx 0.12$ nm. The design of the optics must satisfy the following two conditions to provide this resolution:

1. The spatial resolution of the lens must be finer than the pixel size of the EMCCD camera, which is $x_p = 16$ µm. The focal length of the objective lens we chose (Olympus UPLFLN4x) is 45 mm. To match the focal spot size ($1/e^2$ diameter) to the pixel size, the $1/e^2$ beam diameter at the back aperture of the objective lens should be at least $2f\lambda_0/(\pi x_p/2)) = 2 \times 45$ mm $\times$ 620 nm/($\pi\times$ 8 µm) $\approx$ 2.22 mm, which is smaller than the back aperture diameter (11.7 mm).
2. The angular resolution of the grating must exceed $\Delta\lambda$, which requires the beam to cover at least $\lambda_0/\Delta\lambda = 620$ nm/0.12 nm $\approx$ 5300 grating lines. Since the grating (Ibsen PCG-1908/675-972) has a line density of 1908 lines per mm, this means the minimum $1/e^2$ diameter of the beam on the grating should be at least (5300 lines)/(1908 lines/mm) $\approx$ 2.8 mm. The grating has an area of 20 mm $\times$ 10 mm, sufficiently large for a beam of this size.

In conclusion, to achieve both high spatial and spectral resolution, the beam size must be at least 2.8 mm, which is well within the clear aperture size of the grating and objective lens. The target beam size was designed to be approximately 8 mm, and the actual beam size was slightly smaller than that.

To maximally preserve the quantum properties of the light, all the optics, including routing mirrors, grating, and focusing lens, should have uniformly low loss around 620 nm. The overall quantum efficiency of the spectrometer, including the EMCCD camera quantum efficiency, is measured to be around 75% at 633 nm.

6.2.2 Imaging Resolution

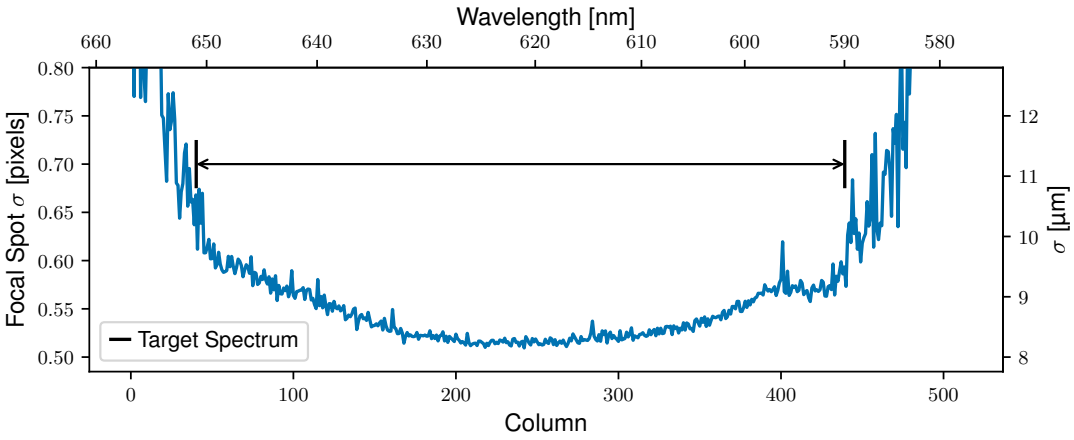


Figure 6.4: **Focal spot size throughout the image plane.** Results of a Gaussian fit to the intensity spread at every column; the standard deviation σ is plotted. The intensity drops outside the target spectrum region, hence the sharp transitions in the fits.

As mentioned in the Discussion, there are imperfections with the detection setup. We do not make full use of the number of pixels: between 10–20% of the spectrometer measures wavelengths outside the target bandwidth. Ideally, customized optics would allow the focal length of the objective lens to match the range of the diffraction angles from the grating to the width of the sensor. As is, the number of detection modes and squeezed modes are close, a possible

cause of decoherence.

Relatedly, the monochromatic focus should be well sub-pixel (16 µm), however, in this experiment we achieved a focus spot with standard deviation $\sigma \approx 0.6$ pixels (Gaussian fit) – on the order of the size of a pixel. We believe the spatial mode quality of the beam is reduced when it is converted, reducing the tightness of the focus. This is also a cause of decoherence because it becomes more difficult to measure the frequency of each photon, and correlations become blurred (Section 6.2.4).

The focal spot size of the spectrometer is shown in Figure 6.4, throughout the wavelength axis (columns) of the camera. The average intensity distribution along the 5 vertical pixels of each column is fit to a Gaussian. The point spread function is a radially symmetric Gaussian in these experiments. It is fairly constant over the image plane (where illuminated), but higher than expected.

6.2.3 Evaluation of the EMCCD Camera

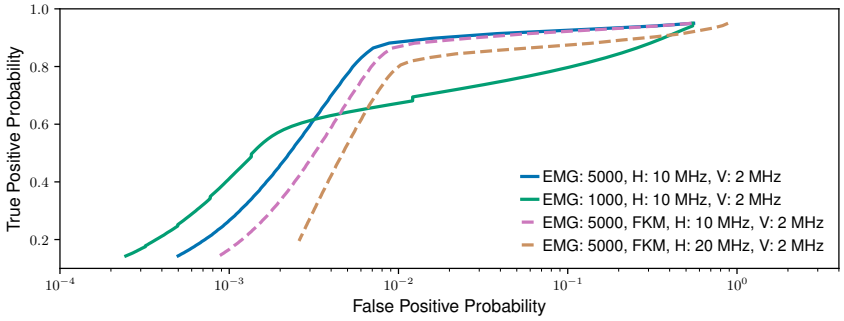


Figure 6.5: **Camera performance. a.** ROC curves for different cameras and operational modes. EMG, EM gain; H, horizontal line rate; V, vertical line rate; FKM, fast kinetic mode.

The NüVü HNü HS 512 EMCCD camera is water-cooled by a thermo-electric chiller (Solid State TCube Edge), and the camera operates with a CCD temperature of -60°C, cooled by the built-in TEC. The camera is mounted on one rotation and two linear stages (Newport RS65; Thorlabs PT1A). One linear stage is used to center the camera with respect to the beam. The other two are actuated by motorized micrometers (Thorlabs Z812B; Newport TRA25CC), in order to align the camera to the objective's focal plane, such that the entire spectrum is in focus when used as a spectrometer. These two degrees of freedom are optimized by minimizing the point spread function of two different wavelengths, alternating until convergence.

The laser repetition rate of the Satsuma is chosen to match the maximum frame-rate of the camera (under a given experimental configuration), and the camera is triggered by the laser (IDQuantique ID900 is used to adjust the delay). The blanking and exposure times are generally set to 0.1 ms, as these values do not adversely affect the frame-rates within the experimental configurations reported.

The camera digital readout ("pixel value", p) is converted to photo-electrons, $\langle n_e \rangle$, by subtracting the bias b and dividing by the total gain. The latter is comprised of the electron multiplication (EM) gain and the analog-to-digital conversion factor k (here 21.43 photoelectrons per pixel unit). Hence $\langle n_e \rangle = kp/g - b$. Bias subtraction is calibrated per pixel where possible, as these exhibited small variations. See Section 6.1.1 for an explanation of the measurements performed with an EMCCD.

In order to quantify the performance of the different configurations of the EMCCD camera, we consider their ROC curves. As discussed in Section 6.1.1,

these represent the trade-off between the photon-detection efficiency (PDE) and the dark count rate: true positive and false positive probabilities. These are shown in Figure 6.5.

6.2.4 Spectrometer POVM and Spectral Discretization

A natural question for our experiment is how to reconcile the fact that we have a continuous basis (frequency), but a discrete set of measurement modes: the spectrometer pixels. Indeed, it is common practice in theory and numerics to discretize fields in the manner:

$$\hat{a}_i = \frac{1}{\omega_i - \omega_{i-1}} \int_{\omega_{i-1}}^{\omega_i} \hat{a}(\omega) d\omega$$

for sufficiently small intervals such that the outcome converges. However, the experimental implications of this procedure are less obvious, especially if the intervals happen to be too large with respect to the spectral features. The correct procedure is to consider the field in the continuous limit and classically accumulate the statistics or probabilities. This coarse-binning effect can therefore be an effective source of decoherence. The discrete case is recovered in the limit where the field properties are constant within the bins.

Additionally, the point spread function when imaging one wavelength onto the pixelated detector must be much smaller than one pixel. This "pixel-arrival" error is otherwise a source of additional decoherence from a classical process.

6.2.5 Multimode Quantum State Sampling

The final ingredient in any quantum-optical computing or sensing protocol is measurement, and in this work we focus on photon-counting measurements. Any quantum advantage for computing or sensing typically scales with the number of photons detected so high average photon numbers and high detection efficiency are crucial. Here we demonstrate sampling of the photon-count distribution of our generated states using an EMCCD camera.

We used 5 rows and all 512 columns of a 512×512 pixel EMCCD camera, whose CCD sensor has a QE of ~95% at 620 nm [199]. The receiver operating characteristic (ROC) curve for thresholding for this camera is shown in Figure 6.6a: this characterizes the false click rate against the photon-detection efficiency (PDE), parametrized by the threshold value (Section 6.1.1). The green marker indicates the threshold used to generate the subsequent plots, resulting in a competitive photon-detection efficiency of ~80%.

Figure 6.6b shows a histogram of the number of photons detected per shot. Blue (left) is prior to thresholding, green (right) is with thresholding. The gain noise in EMCCDs is large, which makes it practically impossible to distinguish between photon numbers ≥1 incident on one pixel. However, statistical averages tend to be accurate, as this is zero mean noise [187] (see Section 6.1.1). Henceforth, we use the term photoelectrons to refer to the amplified charge on the CCD divided by the gain: since the gain is a noisy process, this is not quantized. The first histogram implies an average of almost 700 photons per shot; with thresholding the average number of clicks is just under 500 per shot. The discrepancy is due to both the effective QE and the high rate of multiple photons incident on one pixel.

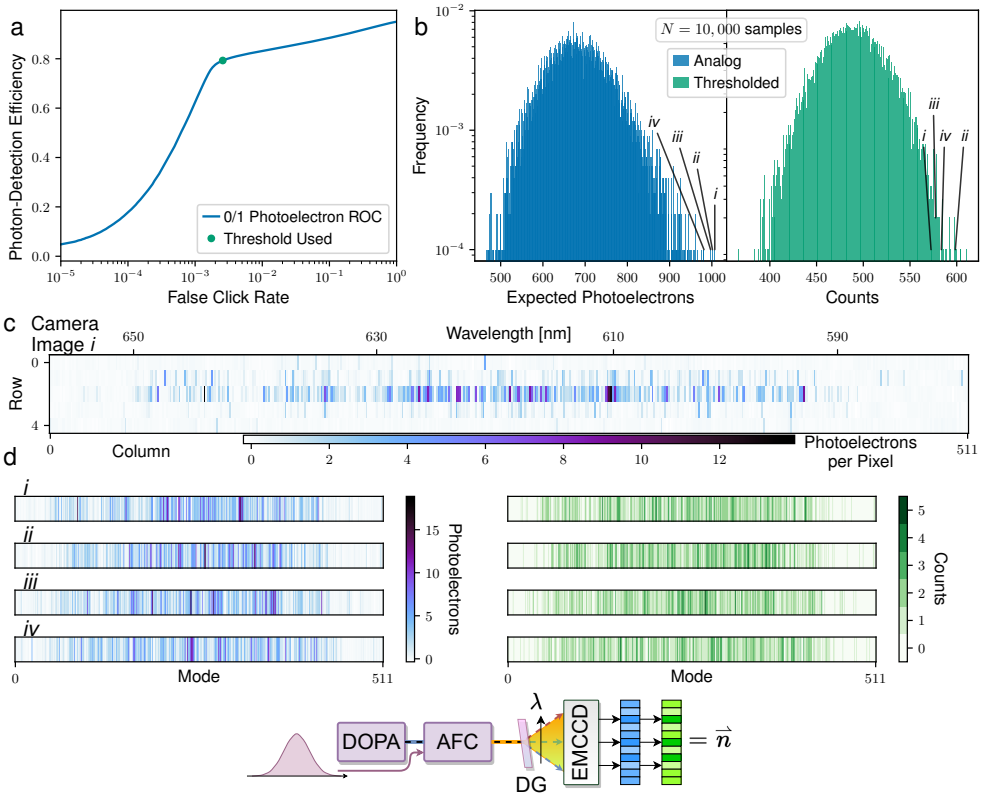


Figure 6.6: **Parallel single-photon detection for multimode-quantum-state measurement: sampling the many-mode, many-photon distribution. a.** Receiver operating characteristic (ROC) curve for the EMCCD camera's detectors. This quantifies the trade-off between false click rate and photon-detection efficiency (PDE). **b.** Histogram of the total photon number per event: analog (noisy) and thresholded. We use the term photoelectrons to refer to the amplified charge on the CCD divided by the gain. Some example, high-photon-number events, are labeled on the histogram with callouts *i*–*iv*; these are referenced in the subsequent subfigures. **c.** Example unprocessed camera image from one sampled event. **d.** Samples integrated vertically (left) and thresholded then summed vertically (right). The experimental schematic (bottom) refers to the experimental configuration for collecting the data shown in **b**–**d**. The data in **a** consists of dark frames collected with a closed shutter (see Sections 6.1.1 and 6.2.3.)

For these frequency-resolving experiments we use more than one pixel-rows of the camera. This is done in order to capture all the photons, due to the spectrometer's point spread function occupying a space larger than one pixel (Gaussian width of $\sigma_{\text{PSF}} \approx 0.6$ pixels). An example of a raw sample is shown in Figure 6.6c, in units of photoelectrons. With this configuration we could sample at a rate of just over 800 Hz. Using a single row would allow an average sampling rate of 15.7 kHz, which is limited by data readout times. Figure 6.6d shows samples integrated vertically, for each frequency bin mode. The point spread function in the vertical direction can allow a thresholding camera to act as a pseudo-photon-number resolving detector, by illuminating more pixels (via astigmatic focus) (an established method [200, 201, 202] used e.g. in Refs [148]). The point spread function in the horizontal dimension, however, acts as an effective decoherence and must be carefully engineered to fit within the dimensions of a pixel (see Section 6.2.4).

6.2.6 Validation of Photon-Counting Cameras for Quantum Advantage

Boson sampling is an intermediate model of linear optical quantum computation [197]. Realizing boson sampling with a level of post-classical computational complexity requires high performance quantum light sources: a large-scale, low-loss photonic circuit; and high-efficiency single-photon detectors – all of which are essential building blocks for universal quantum computation using photons. Gaussian boson sampling (GBS), a variation on this protocol, exploits squeezed vacuum states as input non-classical light sources. For GBS, two

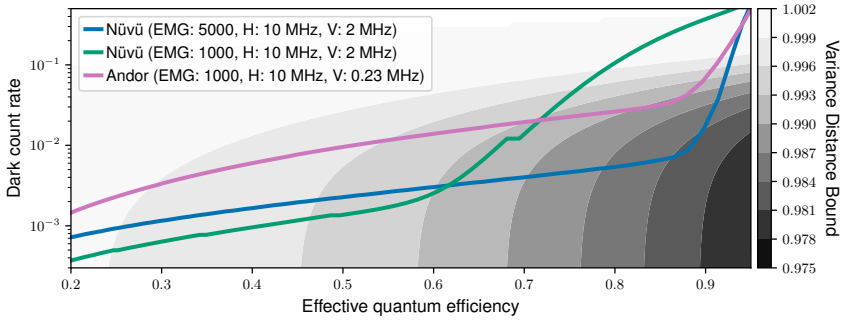


Figure 6.7: **Camera performance vs simulability bound.** EMG, EM gain; H, horizontal line rate; V, vertical line rate. **b.** Comparison to the variance distance assuming an overall transmission of η = 40% and squeezing parameter $r = 1$.

main strategies for classical simulation exist. The first uses the non-negativity of quasi-probability distributions (QPD) (generalizations of the Wigner function) as a strategy for simulation. The second uses the fact that in GBS, the marginal distributions of photon numbers (i.e., the probabilities to observe some subset of detection events irrespective of the others) are informative about the complete probability distribution.

For the QPD based simulations, an inequality exists that that demarcates the "regime of simulability." Thus, any finite-sized experiment must pass this inequality to show that it is not simulable by this strategy. The inequality is given in Ref. [203]:

$$\operatorname{sech}\left\{\frac{1}{2}\Theta\left[\ln\left(\frac{1-2p_D/\eta_D}{\eta e^{-2r}+1-\eta}\right)\right]\right\} > e^{-\epsilon^2/4K},$$

where r is the squeezing parameters, η is the overall photon transmission rate, K is the number of squeezed sources, ϵ is the total variance distance of the experimental GBS samples compared to the ideal cases, Θ is the Heaviside function, η

is the transmission, η_D is the photon detector efficiency and p_D is the dark count rate.

In order to quantify the simulability of the experiment with different photon-counting cameras, we consider the bound for total variance distance ϵ as a function of the parameters determined by the cameras: photon detector efficiency (η_D) and dark count rate (p_D) (as discussed in Section 6.1.1). We estimate the total optical transmission rate to be $\eta \approx 40\%$. In Figure 6.7 we then compare these (η_D, p_D)-plane trajectories to the left-hand side of the above simulability criterion equation. We consider two EMCCD cameras: the NüVü HNü 512 IS and the Andor iXon Ultra 888.

6.3 Highly Multimode Visible Squeezed Light with Programmable Spectral Correlations Through Broadband Up-Conversion

In this section, we demonstrate how to use frequency conversion [204] to enable the use of these visible-light cameras in combination with techniques for strong squeezing possible at longer wavelengths.

6.3.1 Background and Overview

Our demonstration here is an improvement over existing methods of quantum frequency conversion: previous demonstrations are limited to either modest bandwidths or efficiencies [205, 206, 207, 208, 209, 210, 211]. However, adia-

batic frequency conversion (AFC) essentially eliminates this trade-off [212, 213]. We show how this method allows us to obtain robust, efficient and broadband conversion over >45 THz (1390–1750 to 590–650 nm) and near-unity efficiency. Furthermore, it allows unitary control of the multimode entanglement (with, in principle, no additional loss) by manipulation of the complex profile of the broadband pump used to drive the conversion. This architecture provides the best of both worlds: squeezing at telecommunications wavelengths, and photon detection at visible wavelengths. We will show that we are able to generate strong squeezing in over 400 frequency supermodes, resulting in states having a measured mean photon number of nearly 700. By using AFC to efficiently convert the squeezed light to visible wavelengths, and using the highly parallel photon counting made possible by a modern EMCCD, we can directly measure these states. We will also show that we can control the entanglement between different modes by using different spectrally shaped classical fields as the pump of the AFC process, resulting in different measured correlations between photon detections across the frequency modes.

The experimental setup is illustrated in Figure 6.8; an overview is as follows. We use a waveguided degenerate optical parametric amplifier (DOPA) pumped with a pulsed laser: this provides squeezing in a single spatial mode and over many frequency modes (known as the "supermodes"). An adiabatic frequency conversion (AFC) crystal subsequently converts this near-infrared squeezed light to the visible. The temporal profile of the AFC pump pulse is shaped, which controls the conversion process – the linear transformation between infrared and visible light frequencies. Finally we detect the visible squeezed light with an electron-multiplying CCD (EMCCD) camera, serving as an array of high-quality photodetectors. We demonstrate efficient conversion

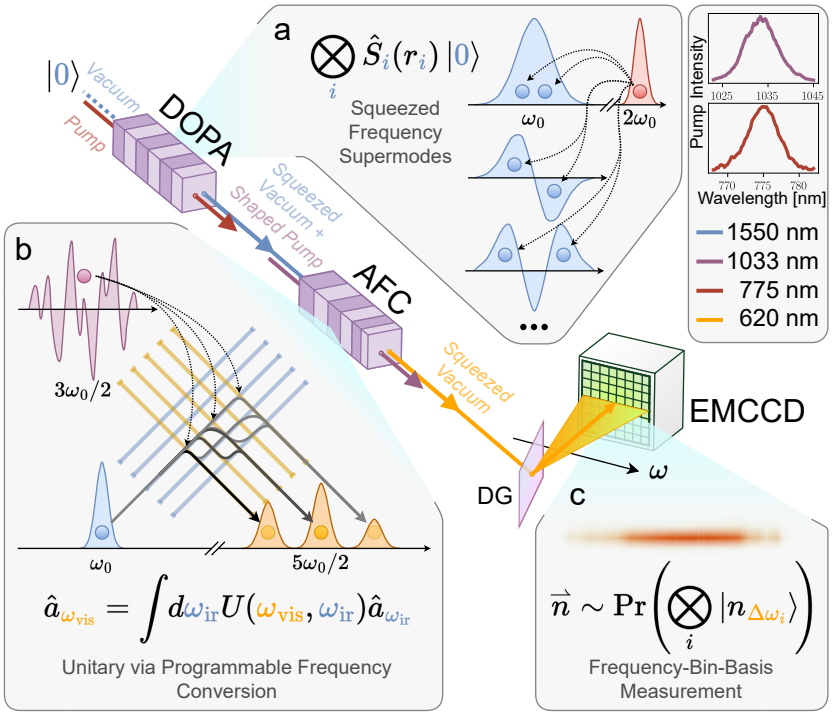


Figure 6.8: **Frequency domain, multimode, visible squeezed state preparation and detection. a.** Highly multimode squeezed vacuum is generated in the degenerate optical parametric amplifier (DOPA). The squeezed modes may be represented by a set of squeezing operators $\hat{S}_i$ acting the vacuum state $|0\rangle$. Each operator squeezes a distinct frequency mode with some squeezing parameter r_i. The squeezing occurs around a central frequency ω_0 equal to half the pump central frequency $2\omega_0$. **b.** Adiabatic frequency conversion (AFC) efficiently converts the squeezed light to visible wavelengths. The pump, a broadband pulse centered at 1033 nm ($3\omega_0/2$), combines with the broadband 1550 nm signal to yield 620 nm ($5\omega_0/2$) light. This operation may be represented as a linear unitary transformation U acting on the infrared and visible fields, represented by the operators $\hat{a}_{\omega_{ir}}$ and $\hat{a}_{\omega_{vis}}$. **c.** The final state, incident on a diffraction grating (DG), is split into frequency modes, and frequency-resolved photon counting is performed with an EMCCD camera. Each measurement yields some photon-number sequence, or vector, n, whose probability distribution depends on the state (determined by $\hat{S}_i$ and U). The camera measures the spectrum in a discrete manner, as the pixels capture the photons within some frequency "bins," denoted $\Delta\omega_i$. Hence, the state's overlap with these bin-basis Fock states, $|n_{\Delta\omega_i}\rangle$, determines the probability distribution.

using AFC. In addition, with the photon-counting camera, we are able to observe spectral photon-number correlations throughout the whole bandwidth at high resolution. With this ability, we generate qualitatively different joint spectra as a proof of concept of frequency-domain unitary transformations by pulse shaping. Please refer to [3] for more details on the DOPA, AFC, pulse shaper, the photon-counting spectrometer, as well as specific experiments.

6.3.2 Upconversion as a Unitary Transformation

Most quantum sensing or computing protocols require unitary operations to be performed on the overall state, because each application requires some specific entanglement structure. Here we demonstrate how the covariance matrix can be transformed by implementing frequency unitaries through the broadband sum frequency generation process. The unitary is applied during the conversion; shaping the AFC pump pulse affects the nonlinear dynamics, changing which wavelengths convert to which. This means that, within certain constraints, we can program the pulse to achieve a desired transformation.

Intuitively, the relative phase of a pump frequency affects the phase of a signal it converts, thereby constructive and destructive interference may promote or suppress the conversion of a signal from one frequency to another, effectively forming a frequency-domain interferometer network. This is illustrated in Figure 6.9a-b.

Some examples of this process and how it influences the covariance matrix are shown in Figure 6.9c. Phase modulations are applied with the pulse shaper, on top of a fixed quadratic phase (chirp) to guarantee a certain pump pulse

duration. The predicted pulse shapes (inferred from the applied phase modulation) are shown for each example. For clarity, we show the correlation rather than covariance matrices, and we subtract a fit to the thermal (classical) part in order to show only the entanglement contribution (the joint spectrum). For reference, Figure 6.9a is derived from the same data as Figure **??**b. As we do not measure the relative phases in the covariance, we focus on transformations in the joint intensity.

In our experiment the bandwidth of the pump (<5 THz) is much smaller than the bandwidth of the squeezed light (~47 THz), which prevents all-to-all coupling (the pump frequencies mediate the change in signal frequency). Additionally, the AFC pump peak intensity is reduced by pulse-shaping, therefore reducing the efficiency. Despite these limitations, it is possible to achieve qualitatively different joint spectra by simply changing the pulse shape.

6.3.3 Influence of Pump Shape on Frequency Conversion

Figure 6.10 demonstrates this effect: by intensity-modulating the pump spectrum (with a fixed phase), for a fixed monochromatic input, we observe that the output signal closely matches the pump profile. However, by virtue of reducing the overall pump energy, the efficiency is affected. In addition, a shift in the input signal ostensibly produces an the same, but shifted, output signal.

While the pump imparts its intensity to the magnitude, the spectral phase of the pump determines the phase in the transformation, which is why we observe that pump phase modulation can generate non-trivial differences in the squeezed-light covariance matrices, in Figure 6.9.

Figure 6.11 shows the simulated conversion process for a broadband signal. The conversion process under an unshaped pump converts fairly uniformly, while the process under a shaped pump is influenced by interference effects, strongly altering the shape of the output. Figure 6.12 shows a series of experimentally-obtained photon spectral correlation matrices as the phase modulation of the AFC pump is gradually varied.

6.4 Discussion

The creation, manipulation, and detection of highly squeezed, highly multimode entangled states are important ingredients of many continuous-variable (CV) schemes for quantum computing, sensing, and communication using photonics. When compared to space or time encoding, frequency encoding has significant advantages for integration and scalability, but it is not always practical to perform unitary transformations on frequency modes without substantial loss. We have experimentally demonstrated efficient, broadband, quantum frequency conversion of highly multimode squeezed light generated in the near infrared into the visible using adiabatic frequency conversion (AFC). This simultaneously allows efficient, parallel photon counting across over 400 squeezed and 500 detection modes using CCD-based photon-detector arrays, and the programming of frequency correlations in the multimode squeezed light. Our approach requires no active phase locking, uses a single optical beam path for the quantum light, and the number of modes and shot rates should be scalable well beyond what we have demonstrated.

To the best of our knowledge, the quantum-optical states we produced

are the largest partially programmable, photon-counted, multimode squeezed states produced by a moderate factor (about 2–4 times larger than previous results, albeit with different limitations and caveats), and by an order of magnitude the largest in the frequency domain. Our work provides a path to constructing large-scale Gaussian boson samplers using frequency encoding. [4]

There are also imperfections with the detection setup, as discussed in Section 6.2.2. Most importantly, the frequency binning is suboptimal, which may be a cause of decoherence. A disadvantage of using an EMCCD camera instead of, for example, an array of superconducting nanowire single-photon detectors for measurement is speed: the camera's frame rate of 800 Hz is slow. However, the EMCCD camera we used has a linerate of 2 MHz (up to 3.33 MHz with higher noise), which is in principle the relevant speed when only using a single row (or few rows) of pixels, as is the case for our measurements. However, there is currently a practical bottleneck to reading out the data, which is limited to line readouts at up to 15.7 kHz on our camera model. Several modern EMCCD cameras, including ours, allow reading out bursts of frames (or regions of whole frames, including lines) at up to MHz rates for short time windows; this could allow fast detection for a limited number of shots in the future. Finally, the photon-number sampling we demonstrated used threshold detection or pseudo-photon-number-resolving measurements; ideally, future imple-

[4]The experiments we report contain the key parts of a Gaussian boson sampler (GBS): squeezing of multiple modes, a unitary, and photon detection. Why then do we not declare our experiments a realization of the world's largest GBS already? The issue is we have imperfect calibration of the squeezed modes and of the unitary transformation realized by AFC, making it unreasonable for us to compare the experimental sampling results we obtained with samples one could obtain from a simulator of our experiment running on a classical computer. Since the simulator would not have a sufficiently accurate description of the modes and the unitary, one wouldn't expect the samples – or the statistics of the samples – to match those from the experiments, which would prevent us from verifying the operation of the GBS in the way that has been done by prior studies [146, 147, 148, 149]. In essence, we have demonstrated a large-scale but poorly calibrated GBS.

mentations could be capable of true photon-number-resolving measurements up to high photon numbers per mode (e.g., using ultra-low-noise CMOS cameras [185, 186]).

Despite these current limitations, the experiment we reported already has some distinct advantages over other platforms. To the best of our knowledge, our experiment marks the first instance of simultaneous sampling with 500 detectors with a similar number of squeezed modes. In comparison to spatial-domain architectures that reach a count of 100s of modes with superconducting detectors, our approach is far less complicated and less expensive to realize for the same number of modes. It appears possible to substantially increase the number of modes and detected photons from what we have already demonstrated, as well as the complexity of the programmed unitaries; crucially, for all these enhancements, the total system transmission is not expected to change significantly, which would be in marked contrast to most space- and time-domain implementations. Overall, we hope our work will enable more widespread study of many-mode, many-photon entangled quantum states, and provide a useful building block for large-scale frequency-encoded CV quantum technologies.

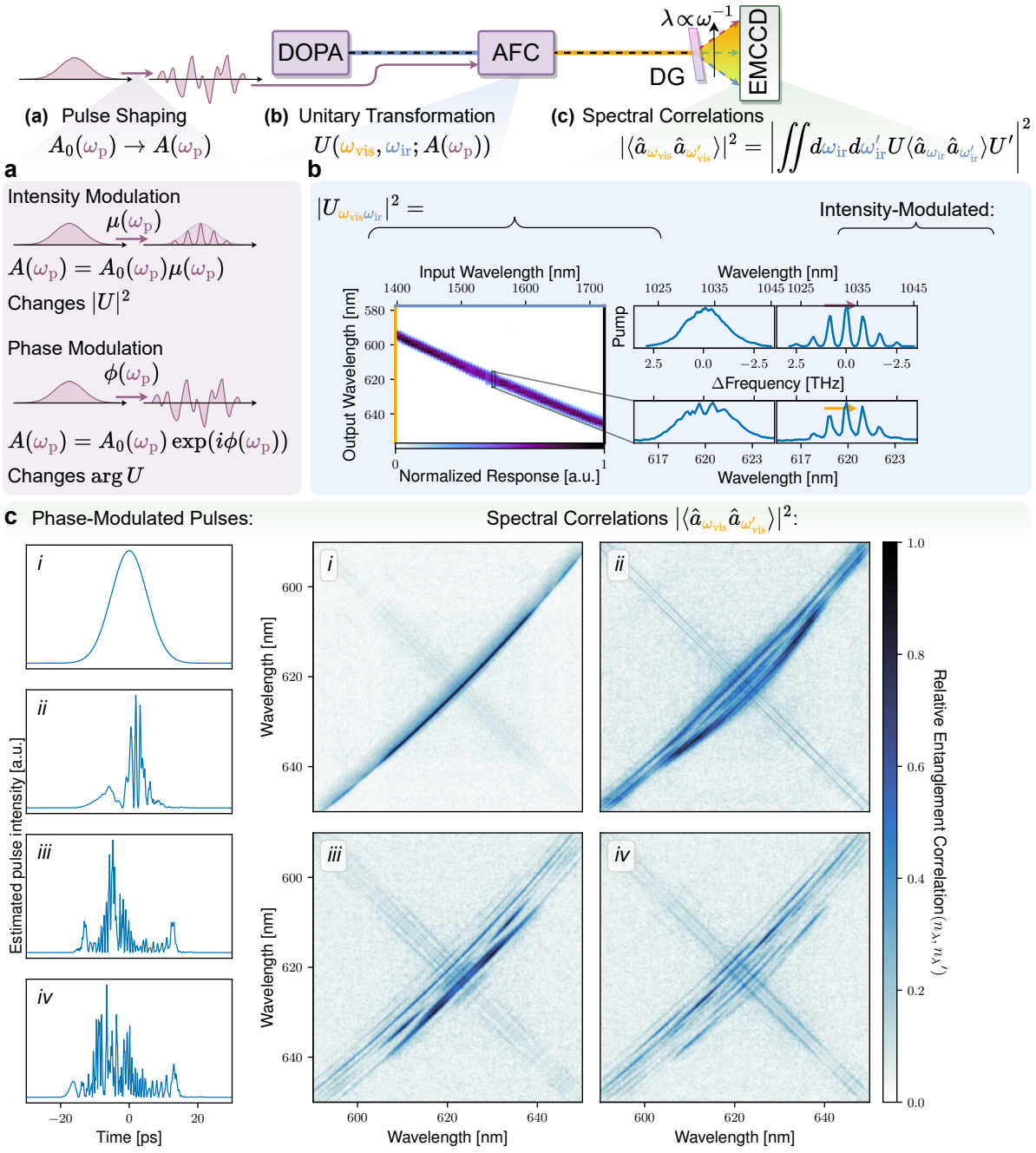

DOPA
AFC
DG
EMCCD
$\lambda \propto \omega^{-1}$
(a) Pulse Shaping
$A_0(\omega_p) \rightarrow A(\omega_p)$
(b) Unitary Transformation
$U(\omega_{vis}, \omega_{ir}; A(\omega_p))$
(c) Spectral Correlations
$|\langle \hat{a}_{\omega_{vis}} \hat{a}_{\omega'_{vis}} \rangle|^2 = \left| \iint d\omega_{ir} d\omega'_{ir} U \langle \hat{a}_{\omega_{ir}} \hat{a}_{\omega'_{ir}} \rangle U' \right|^2$
a
Intensity Modulation
$\mu(\omega_p)$
$A(\omega_p) = A_0(\omega_p)\mu(\omega_p)$
Changes $|U|^2$
Phase Modulation
$\phi(\omega_p)$
$A(\omega_p) = A_0(\omega_p)\exp(i\phi(\omega_p))$
Changes $\arg U$
b
$|U_{\omega_{vis}\omega_{ir}}|^2 =$
Intensity-Modulated:
Input Wavelength [nm]
Output Wavelength [nm]
Normalized Response [a.u.]
Wavelength [nm]
Pump
ΔFrequency [THz]
Wavelength [nm]
c Phase-Modulated Pulses:
Spectral Correlations $|\langle \hat{a}_{\omega_{vis}} \hat{a}_{\omega'_{vis}} \rangle|^2$:
i
ii
iii
iv
Estimated pulse intensity [a.u.]
Time [ps]
Wavelength [nm]
Relative Entanglement Correlation $(n_\lambda, n_{\lambda'})$

Figure 6.9: **Preparing the joint spectrum via frequency conversion.** By pulse shaping the AFC pump $A(\omega_p)$, we can modify the linear unitary transformation U performed by the AFC process, hence the state and the measured spectral photon-number correlations. The pump spectral intensity and phase both play a role in the transformation. **a.** Pulse shaping involves two operations: intensity and phase modulation. The former ($\mu(\omega_p)$) changes the spectral intensity of the pump, which in turn mainly affects the magnitude of elements of U, and the latter ($\phi(\omega_p)$) changes the spectral phase of the pump, which mainly affects the complex phase of the elements of U. **b.** The phase-averaged linear transformation performed by the AFC. The rows of the unitary strongly resemble the pump spectrum, as shown in the subplots: the first pair shows how it compares to the original spectrum, and the second is an example of how it changes given an intensity-modulated pump. The frequency-difference axis is shared among the pump and unitary-row subplots. **c.** These plots show how a phase-modulated pump affects the conversion process and produces more complicated correlation structure. Example spectral intensity correlations are plotted on the right, and the corresponding (inferred) pump intensity profiles on the left. Specifically, we plot the entanglement contribution to photon correlation matrices (correlation matrices with thermal-like components subtracted). The first shows the same data in Figure **??**b, for reference, with no pulse shaping. The next three are the result of random pulse shaping. The corresponding pump intensities, plotted on the right, are not measured directly, but estimated based on the phase modulation applied and the pulse shaper calibration.

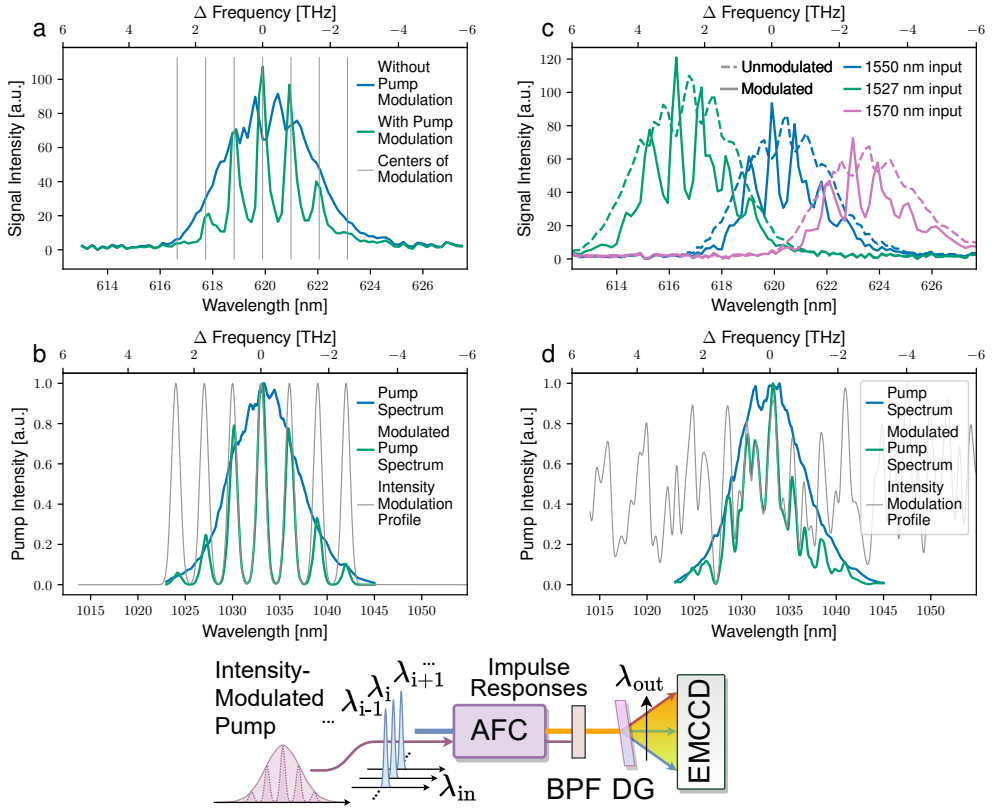


Figure 6.10: **Programming the unitary transformation by programming the pump shape: the effect of the pump spectrum shape on the transformation.** Example of how the pump spectrum affects the frequency conversion profile and linear transformation. **a.** The converted signal spectral profiles, under unmodulated and modulated pumps. **b.** The pump spectra for **a**, unmodulated and under intensity modulation. The conversion profile of the signal closely matches the pump spectrum. **c.** The converted signal spectral profiles, unmodulated and modulated, with three different wavelength inputs. The measured signal intensities are essentially translated in frequency. **d.** The modulated and unmodulated pump spectra for **c**. A random intensity modulation pattern is applied. λ_{in} indicates a monochromatic, infrared input light, and λ_{out} indicates the wavelengths measured by the spectrometer.

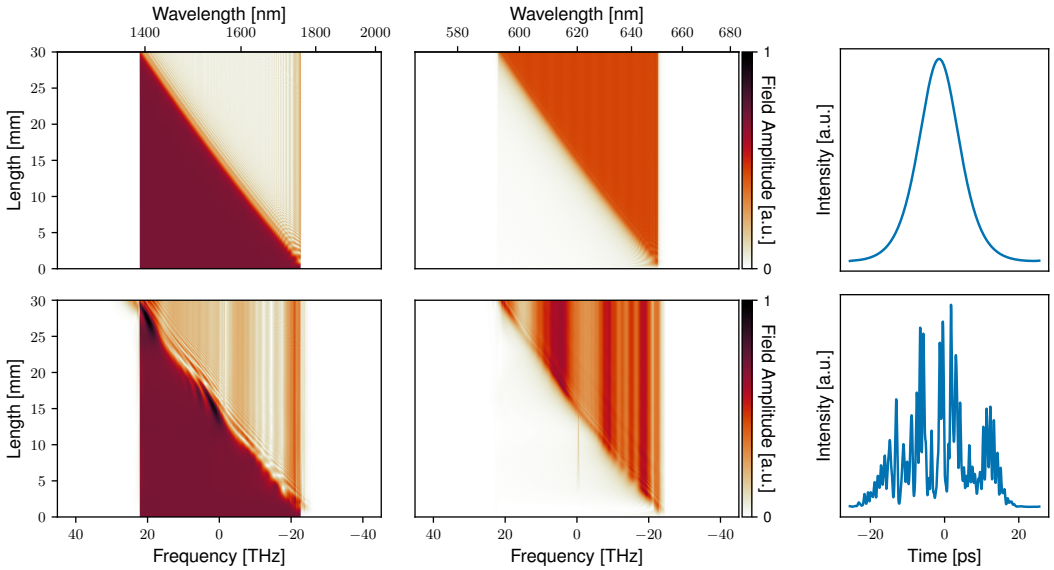


Figure 6.11: **Example conversion dynamics in AFC.** The left panel represents the input modes centered around 1550 nm, the center panel represents the converted modes centered around 620 nm, and the right panel represents the pump profile used in the simulation. **Top:** conversion of a single broadband input with an unshaped pump. **Bottom:** conversion of a single broadband input with an shaped pump. The conversion yields a non-uniform spectrum in the converted wavelength due to interference effects.

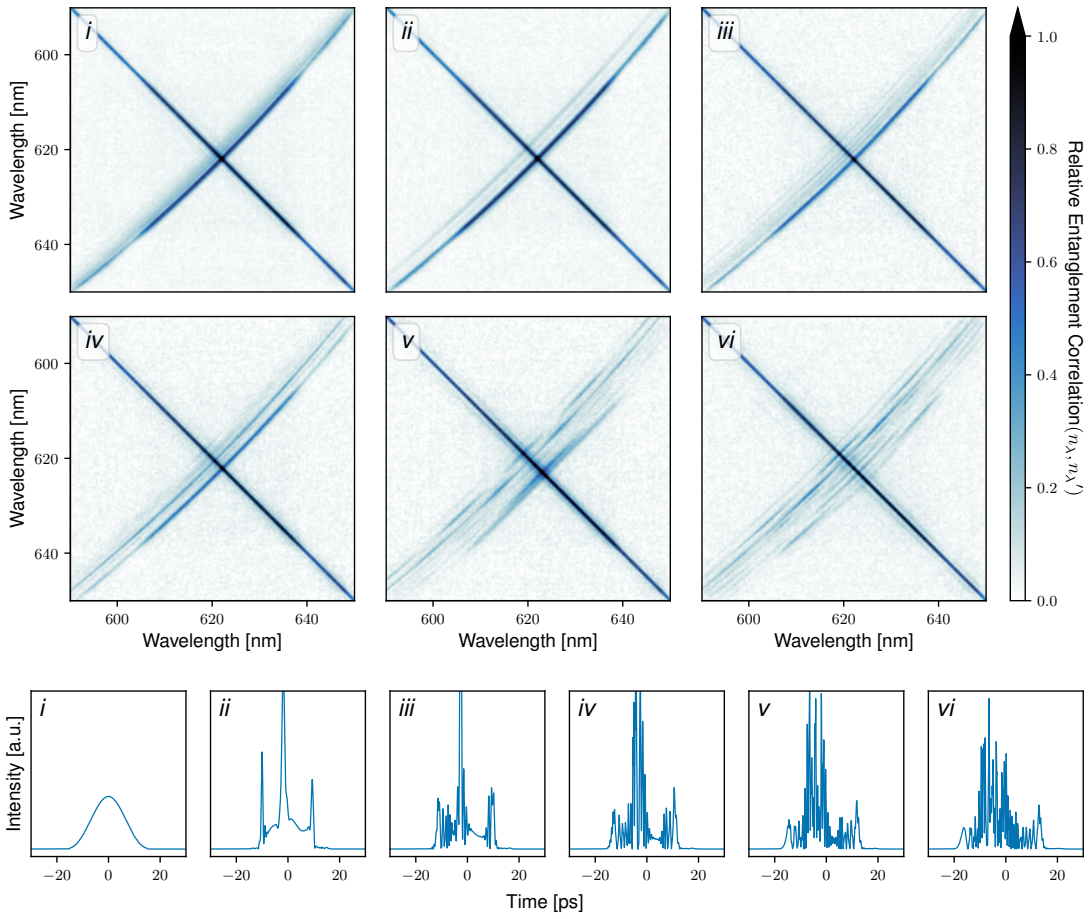


Figure 6.12: **Photon spectral correlations while varying the AFC pump phase modulation.** The top panels are photon number correlation plots, where *i* is the state generated by AFC with a chirped pulse, *vi* with the shaped pulse in Figure 6.9*iv*, and the panels in between are with intermediately-spaced phase modulations of the AFC pump. These six panels show the evolution of the correlation structure as the phase profile varies. The corresponding temporal profiles of the pumps, estimated based on the spectral phase modulations, are shown in the bottom panels. In contrast to Figure 6.9, the diagonal portion of the correlation matrix has not been subtracted.

CHAPTER 7

QUANTUM-LIMITED STOCHASTIC OPTICAL NEURAL NETWORKS OPERATING AT A FEW QUANTA PER ACTIVATION

Much of the content in this chapter is adapted from the work presented in Ma et al. [4].

7.1 Background

Physical systems are subject to various sources of noise. While some noise can be reduced through improvements to the hardware, some noise is fundamentally unavoidable, especially when the system is operated with very little power—which is an engineering goal for neural-network processors. Shot noise is a fundamental noise that arises from the quantized, i.e., discrete, nature of information carriers: the discreteness of energy in the case of photons in optics, and of discreteness of charge in the case of electrons in electronics [214]. A shot-noise-limited measurement of a signal encoded with an average of N_p photons (quanta) will have an SNR that scales as $\sqrt{N_\mathrm{p}}$ [215].[1] To achieve a suitably high SNR, ONNs typically use a large number of quanta for each detected signal. In situations where the optical signal is limited to just a few photons, photodetectors measure and can count individual quanta. Single-photon detectors (SPDs) are highly sensitive detectors that—in the typical *click detector* setting—report, with high fidelity, the absence of a photon (*no click*) or presence of one or more photons (*click*) during a given measurement period [217]. In the regime of an op-

[1] The *shot-noise limit*, which is sometimes also referred to as the *standard quantum limit* [216], can be evaded if, instead of encoding the signal in a thermal or coherent state of light, a quantum state—such as an intensity-squeezed state or a Fock state—is used. In this paper we consider only the case of *classical* states of light for which shot noise is present and the shot-noise limit applies.

tical signal with an average photon number of about 1 impinging on an SPD, the measurement outcome will be highly stochastic, resulting in a very low SNR (of about 1).[2] Conventional noise-aware-training algorithms are not able to achieve high accuracy with this level of uncertainty. Is it possible to operate ONNs in this very stochastic regime and still achieve high accuracy in deterministic classification tasks? The answer is *yes*, and we will show how.

The stochastic operation of neural networks has been extensively studied in computer science as part of the broader field of stochastic computing [218]. In the field of machine learning, binary stochastic neurons (BSNs) have been used to construct stochastic neural networks [219, 220, 221, 222, 223, 224, 225], with training being a major focus of study. Investigations of hardware implementations of stochastic computing neural networks, such as those in Refs. [226, 227] (with many more surveyed in Ref. [228]), have typically been for deterministic complementary metal–oxide–semiconductor (CMOS) electronics, with the stochasticity introduced by random-number generators. While many studies of binary stochastic neural networks have been conducted with standard digital CMOS processors, there have also been proposals to construct them from beyond-CMOS hardware, motivated by the desire to minimize power consumption: direct implementation of binary stochastic neurons using bistable systems that are noisy by design—such as low-barrier magnetic tunnel junctions (MTJs)—has been explored [229, 230, 231], and there have also been proposals to realize hardware stochastic elements for neural networks that could be constructed with noisy CMOS electronics or other physical substrates [232, 233]. ONNs in which noise has been intentionally added [43, 234, 235] have also been studied. Our work with low-photon-count optics is related but distinct from

[2] Again, this is under the assumption that the optical signal is encoded in an optical state that is subject to the shot-noise limit—which is the case for classical states of light.

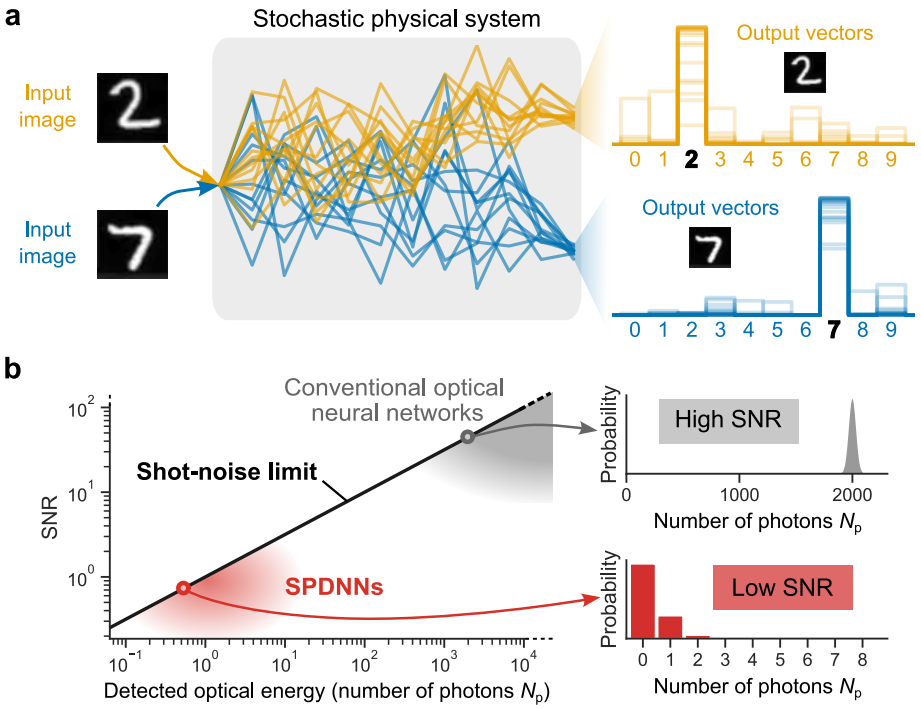


Figure 7.1: **Deterministic inference using noisy neural-network hardware.** **a**, The concept of a stochastic physical neural network performing a classification task. Given a particular input image to classify, repetitions exhibits variation (represented by different traces of the same color), but the class is predicted nearly deterministically. **b**, The single-to-noise ratio (SNR) of single-photon-detection neural networks (SPDNNs) compared to conventional optical neural networks (ONNs). Conventional ONNs operate with high photon budgets (SNR $\gg$ 1) to obtain reliable results, whereas SPDNNs operate with low photon budgets—of up to just a few detected photons per shot (SNR $\sim$ 1). The relation between the detected optical energy (in number of photons N_p) and SNR is SNR = $\sqrt{N_p}$, which is known as the shot-noise limit.

many of the studies cited here in its motivating assumption: instead of desiring noise and stochastic behavior—and purposefully designing devices to have them, we are concerned with situations in which physical devices have large and unavoidable noise but where we would like to nevertheless construct deterministic classifiers using these devices because of their potential for low-energy computing (Figure 7.1).

The **key idea** in this work is to embrace the *inherent uncertainty* of the stochastic physical system, rather than attempting to shape it into a deterministic network with restricted precision. When ONNs are operated in the approximately-1-photon-per-neuron-activation regime and the detectors are SPDs, it is natural to consider the neurons as binary stochastic neurons: the output of an SPD is binary (*click* or *no click*) and fundamentally stochastic. Instead of trying to train the ONN as a deterministic neural network that has very poor numerical precision, one can instead train it as a binary stochastic neural network, adapting some of the methods from the last decade of machine-learning research on stochastic neural networks [222, 223, 224, 236, 225, 237, 228] and using a physics-based model of the stochastic single-photon detection (SPD) process during training. We call this *physics-aware probabilistic modeling*. While a high SNR for the final output is essential for any reliable model outcomes, our approach pushes the boundaries of precision requirements at every step of neural-network propagation (Figure 7.1a). This contrasts with conventional digital devices and noise-mitigation algorithms, which maintain consistently high precision or SNR at every step.

We experimentally implemented a stochastic ONN using as a building block an optical matrix-vector multiplier [1] modified to have SPDs at its output: we

call this a *single-photon-detection neural network* (SPDNN). We present results showing that high classification accuracy can be achieved even when the number of photons per neuron activation is approximately 1, and even without averaging over multiple shots. We also studied in simulation how larger, more sophisticated stochastic ONNs could be constructed and what their performance on CIFAR-10 image classification would be. Apart from the extremely-low SNR, probabilistic modeling also makes our SPDNNs inherently robust to various kinds of imperfections in the setup, including dark counts on the detector, errors in the optical linear operations, fluctuations in the light intensity, etc.

7.2 Physics-Aware Probabilistic Modeling

We consider ONNs in which one or more layers are each constructed from an optical matrix-vector multiplier followed by an array of SPDs (Figure 7.2a–c), and in which the optical powers used are sufficiently low that in each execution of the layer, each SPD has at most only a few photons impinging on it, leading to stochastic measurement outcomes of *no click* or *click*.

In our setting, we aim to perform *inference* using the SPDNN—with its implementation in physical hardware—(Figure 7.2d) and to perform *training* of the SPDNN *in silico* (Figure 7.2e–f). That is, training is performed entirely using standard digital electronic computing.[3]

[3]It is not required that the training be done *in silico* for it to succeed but is just a choice we made in this work. *Hardware-in-the-loop* training, such as used in Ref. [41], is a natural alternative to purely *in silico* training that even can make training easier by relaxing the requirements on how accurate the *in silico* model of the physical hardware process needs to be.

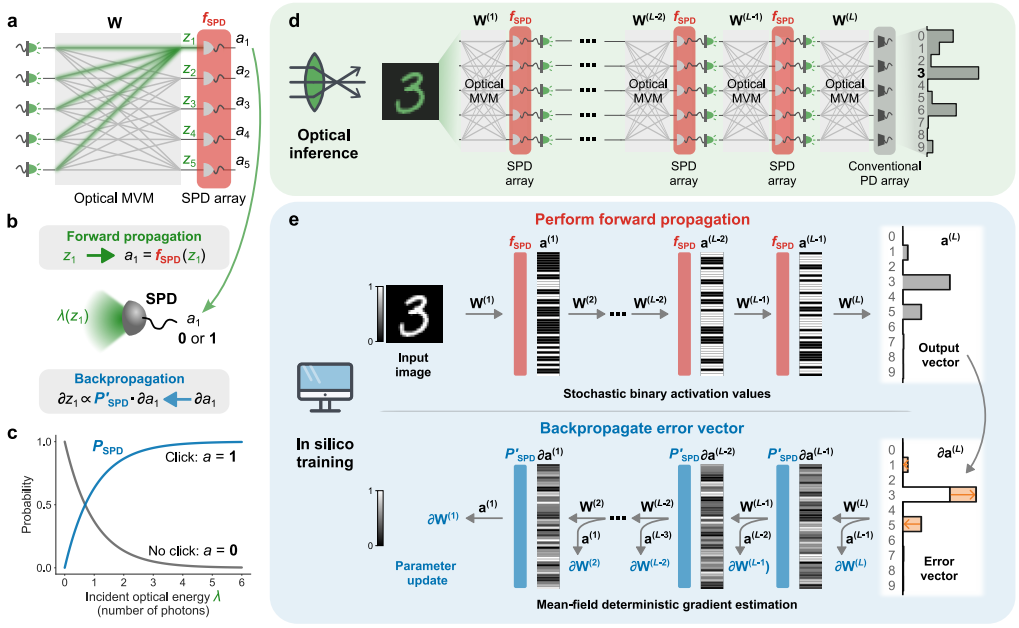


Figure 7.2: **Single-photon-detection neural networks (SPDNNs): *training* and *inference*.** **a**, A single layer of an SPDNN, comprising an optical matrix-vector multiplier (optical MVM, in grey) and single-photon detectors (SPDs; in red). Each output neuron's value is computed by the physical system as $a_i = f_{\text{SPD}}(z_i))$, where z_i is the weighted sum (in green) of the input neurons to the ith output neuron computed as part of the optical MVM, and a_i is the stochastic binary output from a single-photon detector. **b**, Forward and backward propagation through the SPD activation function. The optical energy (λ) incident on an SPD is a function of z_i that depends on the encoding scheme used. Forward propagation uses the stochastic binary activation function f_{SPD}, while backpropagation involves the mean-field function of the probability P_{SPD}. **c**, Probability of an SPD detecting a click (output $a = 1$) or not (output $a = 0$), as a function of the incident light energy λ. **d**, Optical inference using an SPDNN with L layers. The activation values from the SPD array of each layer are passed to light emitters for the optical MVM of the next layer. The last layer uses a conventional photodetector (PD) array instead of an SPD array, and is operated with enough optical energy that the output of this layer has high SNR. **e**, *In silico* training of an SPDNN with L layers. Each forward propagation is stochastic, and during backpropagation, the error vector is passed to the hidden layers using the mean-field probability function P_{SPD} instead of the stochastic activation function f_{SPD}. In this figure, ∂x is shorthand for $\partial C/\partial x$, where C is the cost function.

7.2.1 Training

To train an SPDNN, we perform gradient descent using backpropagation, which involves a forward pass, to compute the current error (or loss) of the network, and a backward pass, which is used to compute the gradient of the loss with respect to the network parameters; our procedure is inspired by backpropagation-based training of stochastic and binary neural networks [222, 225]. We model the forward pass (upper part of Figure 7.2e) through the network as a stochastic process that captures the key physics of SPD of optical signals having Poissonian photon statistics [238]: the measurement outcome of SPD is a binary random variable (*no click* or *click*) that is drawn from the Bernoulli distribution with a probability that depends on the mean photon number of the light impinging on the detector. However, during the backward pass (lower part of Figure 7.2e), we employ a deterministic mean-field estimator to compute the gradients. This approach avoids the stochasticity and binarization of the SPD process, which typically pose difficulties for gradient estimation.

We now give a brief technical description of our forward and backward passes for training; for full details see Appendix A. We denote the neuron preactivations of the lth stochastic layer of an SPDNN as $\mathbf{z}^{(l)} = W^{(l)}\mathbf{a}^{(l-1)}$, where $\mathbf{a}^{(l-1)}$ is the activation vector from the previous layer ($\mathbf{a}^{(0)}$ denotes the input vector $\mathbf{x}$ of the data to be classified). In the physical realization of an SPDNN, $\mathbf{z}^{(l)}$ is encoded optically (for example, in optical intensity) following an optical matrix-vector multiplier (optical MVM, which computes the product between the matrix $W^{(l)}$ and the vector $\mathbf{a}^{(l-1)}$) but before the light impinges on an array of SPDs. We model the action of an SPD with a stochastic activation function, f_{SPD} (Figure 7.2b; Eq. 7.1). The stochastic output of the lth layer is then $\mathbf{a}^{(l)} = f_{\mathrm{SPD}}(\mathbf{z}^{(l)})$.

For an optical signal having mean photon number λ and that obeys Poissonian photon statistics, the probability of a *click* event by an SPD is $P_{\text{SPD}}(\lambda) = 1 - e^{\lambda}$ (Figure 7.2c). We define the stochastic activation function f_{SPD} as follows:

$$f_{\text{SPD}}(z) \coloneqq \begin{cases} 1 & \text{with probability } p = P_{\text{SPD}}(\lambda(z)), \\ 0 & \text{with probability } 1 - p, \end{cases} \tag{7.1}$$

where $\lambda(z)$ is a function mapping a single neuron's pre-activation value to a mean photon number. For an incoherent optical setup where the information is directly encoded in intensity, $\lambda(z) = z$; for a coherent optical setup where the information is encoded in field amplitude and the SPD directly measures the intensity, $\lambda(z) = |z|^2$. In general, the form of $\lambda(z)$ is determined by the signal encoding used in the optical MVM, and the detection scheme following the MVM. We use f_{SPD} in modeling the stochastic behavior of an SPDNN layer in the forward pass. However, during the backward pass, we make a deterministic mean-field approximation of the network: instead of evaluating the stochastic function f_{SPD}, we evaluate $P_{\text{SPD}}(\lambda(z))$ when computing the activations of a layer: $\mathbf{a}^{(l)} = P_{\text{SPD}}(\lambda(\mathbf{z}^{(l)}))$ (Figure 7.2b). This is an adaptation of a standard machine-learning method for computing gradients of stochastic neural networks [222].

7.2.2 Inference

When performing inference (Figure 7.2d), we can run just a single shot of a stochastic layer or we can choose to take the average of multiple shots—trading greater energy and/or time usage for reduced stochasticity. For a single shot, a neuron activation takes on the value $a^{[1]} = a \in \{0, 1\}$; for K shots, $a^{[K]} =$

$\frac{1}{K}\sum_{k=1}^{K} a_k \in \{0, 1/K, 2/K, \ldots, 1\}$. In the limit of infinitely many shots, $K \to \infty$, the activation $a^{[\infty]}$ would converge to the expectation value, $a^{[\infty]} = \mathbb{E}[a] = P_{\mathrm{SPD}}(\lambda(z))$. In this work we focus on the single-shot ($K = 1$) and few-shot $K \leq 5$ regime, since the high-shot $K \gg 100$ regime is very similar to the high-photon-count-per-shot regime that has already been studied in the ONN literature (e.g., in Ref. [1]). An important practical point is that averaging for $K > 1$ shots can be achieved by counting the clicks from each SPD. This is done by integrating several shots of clock cycles that are continuously collected during our experiments. We can think of K as a discrete integration time, so averaging need not involve any data reloading or sophisticated control.

7.3 Numerial Simulation

7.3.1 Incoherent Setup for MNIST Classification

We first evaluated the performance of SPDNNs on the MNIST handwritten-digit-classification benchmark task with a simple, $784 \to N \to 10$ multilayer perceptron (MLP) architecture (Figure 7.3a). Here we use the activation function derived from SPDNNs with an incoherent optical setup (Section A.1). The MNIST dataset has 60,000 images for training and 10,000 images for testing. Each image is grayscale and has $28 \times 28 = 784$ pixels. To meet the non-negative encoding of incoherent light, the input images are normalized to have pixel values of range 0 to 1.

The models consist of two linear layers: the $784 \to N$ hidden layer has the weight matrix $W^{(1)}$ with a shape of $N \times 784$, and the $N \to 10$ output layer has the

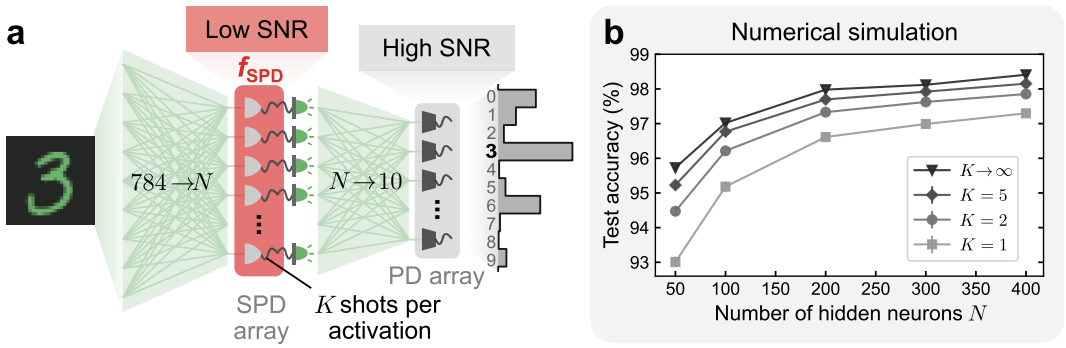


Figure 7.3: **Incoherent SPDNNs on MNIST handwritten-digit classification.** **a**, An SPDNN realizing a multilayer perceptron (MLP) architecture of N neurons in the hidden layer. The hidden layer ($784 \to N$) was computed using an incoherent optical matrix-vector-multiplier (MVM) followed by a single-photon-detector (SPD) array. Each SPD realized a stochastic activation function for a single hidden-layer neuron. During a single inference, the hidden layer was executed a small number of times ($1 \leq K \leq 5$), yielding averaged activation values. The output layer ($N \to 10$) was realized either optically—using an optical MVM and high photon budget to achieve high readout SNR[6], as in conventional ONNs, or with a digital electronic processor, yielding a result with full numerical precision. **b**, Simulated test accuracy of MNIST handwritten-digit classification for models with different numbers of hidden neurons N and shots per activation K.[7] Each activation value is obtained by averaging K shots of stochastic binary SPD readouts. When $K \to \infty$, the stochastic activations a_i become the expectations $\mathbb{E}[a_i]$, which are deterministic. The test accuracy with few shots is close to the accuracy achieved in the deterministic limit.

weight matrix $W^{(2)}$ with a shape of $10 \times N$. The SPD activation function is applied to each hidden neuron after the linear operation of $W^{(1)}$ to compute the neuron activations; then the computed activation values are passed to the output layer to produce the output vectors. The elements in the first linear operation, $W^{(1)}$, are clamped to be non-negative to meet the requirement of an incoherent optical setup. In general, real-valued weights can be realized with an incoherent optical

MVM if some digital-electronic post-processing is available. In our case, where the activations are measured by SPDs, the activation function is directly applied in the single-photon detection process, which makes digital post-processing impossible. Similarly, biases of the linear operations are also disabled. If we want to get away with digital post-processing by applying a bias term directly to the optical intensity, at the level of a few photons, the approach is also challenging in experiments. However, since the output layer is implemented using conventional optical computing with a higher signal-to-noise ratio (SNR), we can effectively implement real-valued weights of $W^{(2)}$. In optical implementation, this would involve extra operations to map these values onto the incoherent setup.

During the training process, we apply the LogSoftmax function to the output vectors and use the cross-entropy loss to construct the loss function. To avoid the issue of vanishing gradients, we clamp the pre-activation values at $\lambda_{\text{max}} = 3$ photons. It is important to note that due to the stochastic nature of the neural networks, each forward pass generates different output values, even with the same weights and inputs. However, we only use a single forward pass in each training epoch, which has been shown to be the most efficient training approach. The stochasticity introduced in each forward propagation could add to the random search of the stochastic optimizer itself, helping with the training process.

We have found that using the SGD optimizer [239] with small learning rates leads to better accuracy compared to other optimizers, such as AdamW [240]. Although training with SGD takes longer overall, it helps us achieve a better-optimized model in the end. For our final results, we used a batch size of 128

and a learning rates of 0.001 for the hidden layer and 0.01 for the output layer in the SGD optimizer. We trained each SPDNN model for 10,000 epochs to obtain optimized parameters, and an even higher number of epochs may be needed to achieve better accuracy. Given the small learning rate and the significant amount of noise in the model, the number of epochs required is much larger than what is typically seen in common neural network training processes. The training and test errors for an incoherent MLP SPDNN with a structure of $784 \to 400 \to 10$ are shown in Figure 7.4. The training process was performed on a GPU (Tesla V100-PCIE-32GB) and took approximately eight hours to complete.

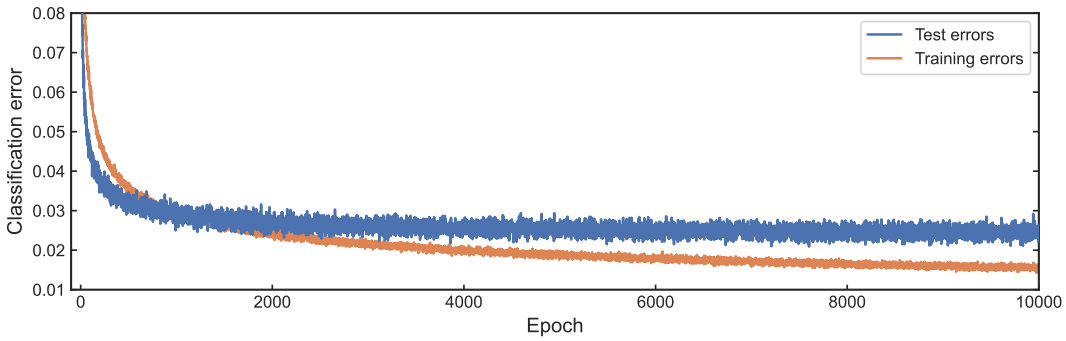


Figure 7.4: **Training curves of an incoherent SPDNN model for MNIST classification.** The plot illustrates the progression of test and training errors throughout the training process of an incoherent SPDNN model with an MLP architecture of $784 \to 400 \to 10$. The optimization is conducted using an SGD optimizer with learning rates of 0.001 for the hidden layer and 0.01 for the output layer. The final trained model is obtained at 10,000 epochs.

The magnitude of the weight element values in the first linear layer, $W^{(1)}$, is influenced by the range of values in the input vectors, $a^{(0)}$, and the specific form of the SPD activation function, $f_{\text{SPD}}^{\text{Incoh}}(z)$. In the forward pass of an incoherent SPDNN, the pre-activation values, $z^{(1)}$, are computed as $z^{(1)} = a^{(0)}W^{(1)\top}$. The activation function, $f_{\text{SPD}}^{\text{Incoh}}(z)$, is defined as $\mathbf{1}_{t<P_{\text{SPD}}(z)}$, where t is a random variable uniformly distributed between 0 and 1, and $P_{\text{SPD}}(z)$ represents the probability of

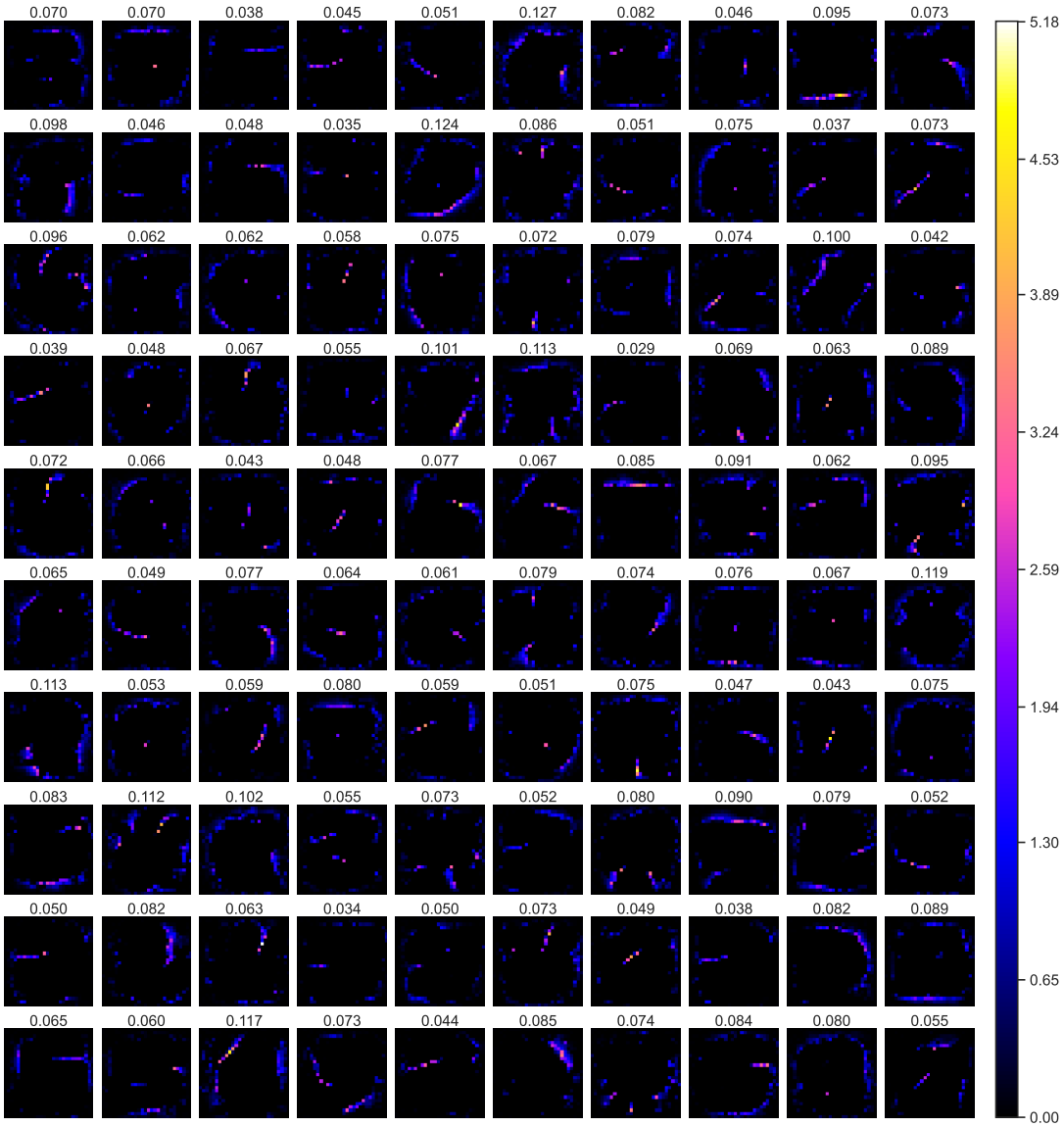


Figure 7.5: **Visualization of weight elements in the first linear layer of an incoherent SPDNN.** The architecture of this model is $784 \to 100 \to 10$, and we display the weight matrix $W^{(1)}$ of the first layer (with dimensions 100×784). Each block represents a row vector in $W^{(1)}$ containing 784 elements. These column vectors are rearranged to form a 2D block with dimensions 28×28, matching the original shape of the MNIST input images. The 100 rows in $W^{(1)}$, corresponding to the 100 hidden neurons in the neural network, are arranged in a 10×10 grid to be visualized. The average value of each block is indicated at the top, and the overall average value of the weight matrix is ~ 0.07.

photon detection. When the input vectors, $a^{(0)}$, are normalized to the range of 0 to 1, the weight elements in $W^{(1)}$ are optimized based on the specific form of P_{SPD} because it depends on the exact value of pre-activations z. In our simulation of an incoherent SPDNN, the elements of $z^{(1)}$ are represented in terms of photon numbers, where the value 1 corresponds to 1 photon. When $z \gtrsim 3$, P_{SPD} reaches the plateau part of the probability function. Thus, we have to make sure the value of the pre-activation to be around 1 photon to ensure an effective forward pass. When considering a uniform bright image where each element has the maximum value of 1, and with an input vector size of $28 \times 28 = 784$, if we aim for an output value of approximately 1 photon, the average value of the weight elements in $W^{(1)}$ should be around $1/784 \approx 0.0013$. The average pixel value in the MNIST dataset is approximately 0.13 (when each pixel value is normalized to the range of 0 to 1). Based on this, we can estimate that to achieve an output value of approximately 1 photon, the average weight element value should be around 0.01. Taking into account that both the input images and weight matrices tend to be sparse, this estimation may be slightly lower than the actual scenario. Figure 7.5 illustrates the matrix elements of $W^{(1)}$ for a model with $N = 100$ hidden neurons. The weight elements range from 0 to 5.18, with an average value of 0.07. Each block represents a row vector of size 784, rearranged in the form of 28×28. The average value of $W^{(1)}$ may vary slightly in different network structures, ranging from 0.06 to 0.08.

During the inference of SPDNNs, the pixel values of the test images are normalized to the range of 0 to 1 as well. This will be correspond to the dynamic range on the optical setup. We trained incoherent MLP-SPDNN models with varying numbers of hidden neurons, N, ranging from 10 to 400. As discussed in Section A.1, we can adjust the number of SPD measurements per activation,

denoted as K, to control the level of stochasticity in the models.

Model	$K = 1$	$K = 2$	$K = 3$	$K = 5$	$K = 7$	$K = 10$	$K \to \infty$
784–10–10	78.03 ± 0.32	83.18 ± 0.26	84.79 ± 0.22	86.13 ± 0.17	86.65 ± 0.17	87.08 ± 0.16	87.91 ± 0.00
784–20–10	86.74 ± 0.24	89.98 ± 0.18	90.96 ± 0.15	91.71 ± 0.13	92.00 ± 0.13	92.22 ± 0.13	92.66 ± 0.00
784–50–10	93.04 ± 0.16	94.49 ± 0.15	94.92 ± 0.12	95.24 ± 0.11	95.38 ± 0.10	95.47 ± 0.09	95.73 ± 0.00
784–100–10	95.20 ± 0.16	96.24 ± 0.11	96.53 ± 0.10	96.75 ± 0.09	96.85 ± 0.07	96.91 ± 0.07	97.02 ± 0.00
784–200–10	96.62 ± 0.12	97.33 ± 0.10	97.54 ± 0.08	97.70 ± 0.08	97.75 ± 0.08	97.80 ± 0.06	97.98 ± 0.00
784–300–10	97.00 ± 0.12	97.61 ± 0.08	97.80 ± 0.08	97.93 ± 0.07	97.97 ± 0.06	98.01 ± 0.05	98.12 ± 0.00
784–400–10	97.31 ± 0.11	97.85 ± 0.10	98.01 ± 0.09	98.15 ± 0.06	98.20 ± 0.06	98.27 ± 0.05	98.41 ± 0.00

Table 7.1: **Test accuracy (%) of incoherent MLP-SPDNNs on MNIST with varying hidden layer size N and shots per activation K.** These models have an MLP structure of $784 \to N \to 10$, where N represents the number of hidden neurons. Each hidden neuron uses K shots of binary SPD readouts to compute its activation value. The reported test accuracy values are obtained by calculating the mean value and standard deviation over 100 repetitions of inferences on the MNIST test set, which comprises 10,000 images.

The results of the MNIST test accuracy for different combinations of N and K are summarized in Table 7.1. The values of N include 10, 20, 50, 100, 200, 300, and 400, while K takes on the values of 1, 2, 3, 5, 7, 10, and ∞. In the case of $K \to \infty$, we use the expectation of the activation values, P_{SPD}, as the activation function, which is equivalent to integrating an infinite number of shots per SPD detection. This serves as an upper bound that is approached as K increases. Due to the stochastic nature of SPDNNs, the output vectors vary across different repetitions of inference. To capture the overall behavior of the models, we repeated the full inference process 100 times for each structure with N hidden neurons and K shots per activation. This allows us to calculate the mean test accuracy and standard deviation, representing the distribution of test accuracies. Each independent repetition of inference uses the MNIST test dataset, consisting of 10,000 images. We observe that as either N or K increases, the mean test accuracy tends to improve while the standard deviation decreases.

7.3.2 Coherent Setup with More Complex Architecture

We have successfully demonstrated two-layer SPDNNs that are compatible to the incoherent optical setups as described in Chapter 3, but can SPDNNs be used to implement deeper and more sophisticated models? One of the limitations of our experimental apparatus was that it used an intensity encoding with incoherent light and as a result could natively only perform operations with non-negative numbers. In this section we will show that SPDNNs capable of implementing signed numbers can be used to realize multilayer models (with up to 6 layers), including models with more sophisticated architectures than multilayer perceptrons—such as models with convolutional layers.

ONNs based on coherent light can naturally encode sign information in the phase of the light and have been realized in many different physical platforms [12, 57, 13, 56, 63, 17, 16]. We propose—and study in simulation—SPDNNs using coherent light. Neuron values are encoded in optical amplitudes that are constrained to have phases that are either 0 (positive values) or π (negative values). With this encoding, detection by an SPD—which measures intensity and is hence insensitive to phase—results in a stochastic nonlinear activation function that is symmetric about zero (Figure 7.6a). Alternative detection schemes could be employed that would modify the activation function, but we have focused on demonstrating the capabilities of this straightforward case, avoiding introducing additional experimental complexity.

We performed two sets of simulation experiments: one on coherent SPDNNs trained to perform MNIST handwritten-digit classification (Figure 7.6d), and one on coherent SPDNNs trained to performed CIFAR-10 image classification (Figures 7.6e).

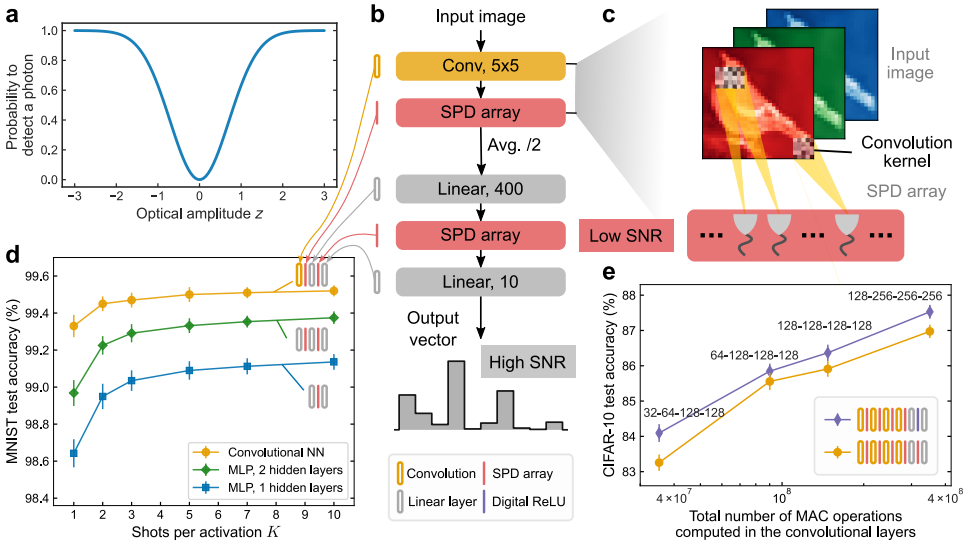


Figure 7.6: **Performance of *coherent* single-photon-detection neural networks (SPDNNs). a**, The probability of detecting a photon as a function of the input light amplitude in a coherent SPDNN. Real-valued numbers are encoded in coherent light with either 0 phase (positive numbers) or π phase (negative numbers). Measurement by a single-photon detector (SPD) results in the probabilistic detection of a photon that is proportional to the square of the encoded value z, in comparison to intensity encodings with incoherent light. **b**, Structure of a convolutional SPDNN with a kernel size of 5×5. Single-shot SPD measurements ($K = 1$) are performed after each layer (by an SPD array), except for the output layer. Average 2×2 pooling is applied after each convolutional operation. A digital rectified linear unit (ReLU) activation function can also be used in the linear layer as an alternative. **c**, Schematic of a convolutional layer with SPD activations. **d**, Simulated test accuracy of coherent SPDNNs with varying architecture performing MNIST handwritten-digit classification. The multilayer perceptron (MLP) models had 400 neurons in each hidden layer. The convolutional model consisted of a convolutional layer with 16 output channels, followed by two linear layers with an SPD activation inbetween. **e**, Simulated test accuracy of coherent SPDNNs with varying architecture performing CIFAR-10 image classification. The models have four convolutional layers, each followed by SPD activation functions. The two linear layers can either be implemented in full-precision with a ReLU activation function (in purple) or using the SPD activation function. The number of output channels for each convolutional layer is indicated above the corresponding data point.

MNIST classification

The MNIST handwritten-digit classification task was performed using the same simulation configurations as with incoherent SPDNNs, but but using the coherent SPD activation function and real-number operations. Unlike the previous case, no clamping of the weights was necessary. The models were trained using the SGD optimizer with a learning rate of 0.01 for the hidden layers and 0.001 for the last linear layer, for a period of 10,000 epochs. To evaluate the impact of model size, we trained models with both one and two hidden layers. The training curves of the model with the structure of $784 \to 400 \to 400 \to 10$ are shown in Figure 7.7. The results for models with different structures and shots of SPD measurements per activation can be found in Table 7.2, and the weights of a model with the structure of $784 \to 100 \to 10$ are illustrated in Figure 7.8.

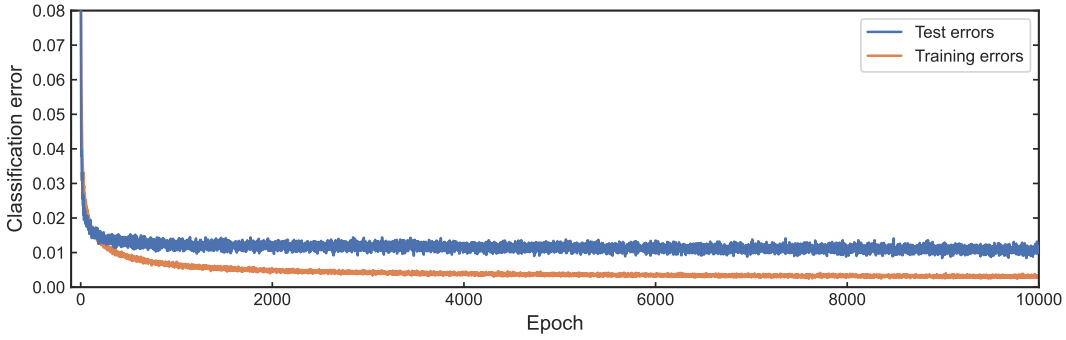


Figure 7.7: **Training curves of a coherent SPDNN model on MNIST classification.** The plot shows the evolution of the test and training errors during the training process of a coherent SPDNN model with an MLP architecture of $784 \to 400 \to 400 \to 10$ neurons. The optimization is performed using SGD with different learning rates for the hidden (0.001) and output (0.01) layers. The end result of the training is represented by the final values of the 10,000 epochs.

Furthermore, convolutional SPDNNs were also used for MNIST classifica-

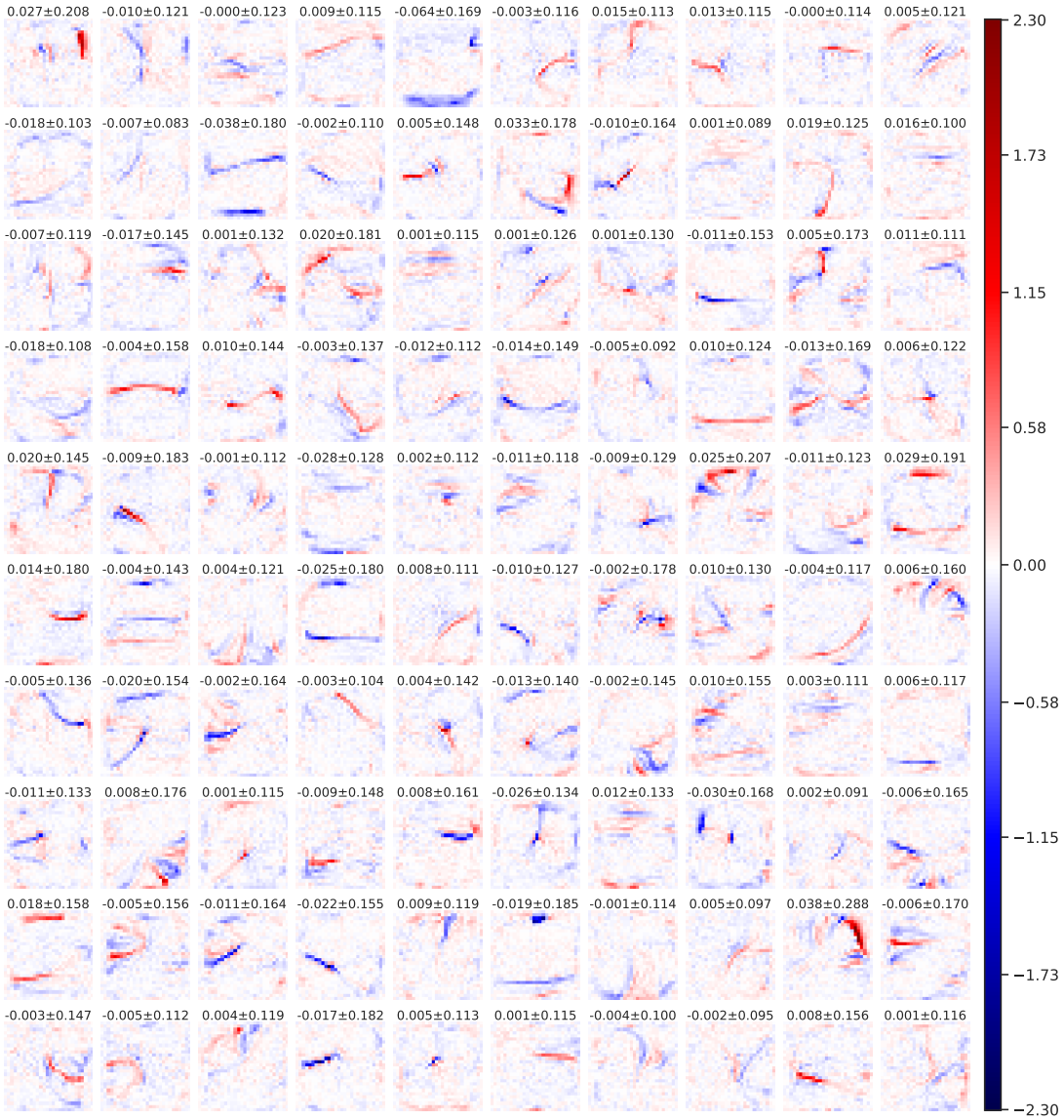


Figure 7.8: **Visualization of weight elements in the first linear layer of a coherent SPDNN.** The architecture of this model is $784 \to 100 \to 10$, and we display the weight matrix $W^{(1)}$ of the first layer (with dimensions 100×784). Each block represents a row vector in $W^{(1)}$ containing 784 elements. These column vectors are rearranged to form a 2D block with dimensions 28×28, matching the original shape of the MNIST input images. The 100 rows in $W^{(1)}$, corresponding to the 100 hidden neurons in the neural network, are arranged in a 10×10 grid to be visualized. The average value and standard deviation of the elements in each block are indicated at the top.

Model	$K = 1$	$K = 2$	$K = 3$	$K = 5$	$K = 7$	$K = 10$	$K \to \infty$
784–10–10	80.21 ± 0.27	85.63 ± 0.24	87.49 ± 0.19	88.85 ± 0.15	89.42 ± 0.15	89.87 ± 0.14	90.70 ± 0.00
784–20–10	89.31 ± 0.23	92.43 ± 0.17	93.42 ± 0.14	94.10 ± 0.13	94.39 ± 0.14	94.60 ± 0.11	95.05 ± 0.00
784–25–10	91.51 ± 0.20	94.16 ± 0.16	94.89 ± 0.14	95.47 ± 0.12	95.70 ± 0.10	95.86 ± 0.08	96.10 ± 0.00
784–30–10	92.72 ± 0.19	94.92 ± 0.13	95.58 ± 0.12	96.05 ± 0.10	96.26 ± 0.09	96.41 ± 0.08	96.74 ± 0.00
784–50–10	95.50 ± 0.15	96.88 ± 0.11	97.26 ± 0.09	97.55 ± 0.08	97.67 ± 0.07	97.77 ± 0.07	97.93 ± 0.00
784–100–10	97.41 ± 0.13	98.16 ± 0.08	98.36 ± 0.08	98.52 ± 0.07	98.58 ± 0.06	98.61 ± 0.05	98.70 ± 0.00
784–200–10	98.34 ± 0.10	98.76 ± 0.07	98.88 ± 0.07	98.97 ± 0.05	99.00 ± 0.05	99.04 ± 0.04	99.12 ± 0.00
784–400–10	98.64 ± 0.08	98.95 ± 0.06	99.04 ± 0.05	99.09 ± 0.05	99.12 ± 0.04	99.14 ± 0.03	99.19 ± 0.00
784–400–400–10	98.95 ± 0.08	99.21 ± 0.05	99.29 ± 0.04	99.33 ± 0.04	99.35 ± 0.04	99.37 ± 0.03	99.40 ± 0.00
784–C16–400–10	99.33 ± 0.06	99.45 ± 0.04	99.47 ± 0.04	99.50 ± 0.04	99.51 ± 0.03	99.52 ± 0.03	99.54 ± 0.00

Table 7.2: **Test accuracy (%) of coherent SPDNN models on MNIST with different model structures and shots per activation K.** These models have an MLP structure with one or two hidden layers with the number of neurons denoted in the table, except for the last model, which has a convolutional layer denoted as "C16" with 16 output channels and followed by a 2×2 average pooling and the following linear layers. The mean accuracy and standard deviation are calculated based on 100 repetitions of inferences using the MNIST test set of 10,000 images.

tion. The architecture included a convolutional layer with 16 output channels, a kernel size of 5×5 and a stride of 1. An SPD activation function was immediately applied after each convolution layer, without batch normalization. Average pooling of 2×2 was performed after each of the SPD activations. After the convolution layer, the total number of features was 3136, then the convolution layers were followed by a linear model of $3136 \to 400 \to 10$, with the SPD activation function applied at each of the 400 hidden neurons as well. This structure is depicted in Figure 7.6b. For optimization, we used an SGD optimizer with a learning rate of 0.01 for the entire model. The convolutional SPDNN model can be optimized easily without fine-tuning the parameters. After 200 epochs, the accuracy quickly reached 99.4%.

CIFAR-10 classification

Figure 7.6e shows the results of simulating variants of a 6-layer convolutional SPDNN (comprising 4 convolutional layers and 2 fully connected, linear layers) on CIFAR-10 image classification. All these simulation results were obtained in the single-shot ($K = 1$) regime. The number of channels in each convolution layer was varied, which affects the total number of MACs used to perform an inference. We observed that the test accuracy increased with the size of the SPDNN, with accuracies approaching those of conventional convolutional neural networks of comparable size [241], as well as of binarized convolutional neural networks [225, 242, 243]. In the models we simulated that only used SPD as the activation function (i.e., the ones in which there are no 'Digital ReLU' blocks), the high-SNR linear output layer had only 4000 MAC operations, so the number of MACs in the high-SNR layer comprises less than 0.01% of the total MACs performed during an inference. The models we simulated are thus sufficiently large that the total optical energy cost would be dominated by the (low-SNR) layers prior to the (high-SNR) output layer. Equivalently, the optical energy cost per MAC would be predominantly determined by the cost of the low-SNR layers. These simulation results illustrate the ability of SPDNNs to scale to larger and deeper models, enabling them to perform more challenging tasks. The symmetric stochastic activation function that is realized by SPD of coherently encoded real values yields good accuracies on both MNIST and CIFAR-10 benchmarks and is straightforward to implement experimentally.

The CIFAR-10 dataset [244] has 60,000 images, each having $3 \times 32 \times 32$ pixels with 3 color channels, that belong to 10 different categories, representing airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships and trucks.

The dataset is partitioned into a training set with 50,000 images and a test set with 10,000 images. The pixel values have been normalized using the mean value of $(0.4914, 0.4822, 0.4465)$ and standard deviation of $(0.2471, 0.2435, 0.2616)$ for each of the color channels. To boost performance, data augmentation techniques including random horizontal flips (50% probability) and random 32×32 crops (with 4-pixel padding) were implemented during training. We used the AdamW optimizer [240] with a learning rate of 0.0005 and betas of $(0.99, 0.98)$. The models were trained for thousands of epochs.

The convolutional SPDNNs have a structure where the SPD activation function is applied after each convolution layer and before an average pooling of 2×2. The final architecture consists of a series of convolution layers followed by a linear layer of 400 neurons, and a final layer of $400 \rightarrow 10$ for the output. Similar to the convolutional models trained for MNIST, the convolutional layers use a kernel size of 5×5, a stride size of 1 and padding of 2. Batch normalization was used in the models after each convolutional layer. Either SPD or ReLU activation function was applied to each of the 400 neurons in the first linear layer, as depicted in Figure 7.6e.

After N_{conv} convolutional layers ($N_{\text{conv}} = 2$, 3 or 4 in this case) with the number of output channels of the last one to be $N_{\text{chan}}^{\text{last}}$ (either 128 or 256 in this case), the feature map of $(32/2^{N_{\text{conv}}})^2 \times N_{\text{chan}}^{\text{last}}$ is flattened to a vector, followed by two linear layers of $(32/2^{N_{\text{conv}}})^2 N_{\text{chan}}^{\text{last}} \rightarrow 400 \rightarrow 10$. In addition to the results presented in Figure 7.6e, we experimented with more architectures ranging from 2 to 4 convolution layers, and the results are displayed in Figure 7.9. In these models, only the SPD activation function was used. The x-axis of the plot represents the number of multiply–accumulate (MAC) operations in the convolutional lay-

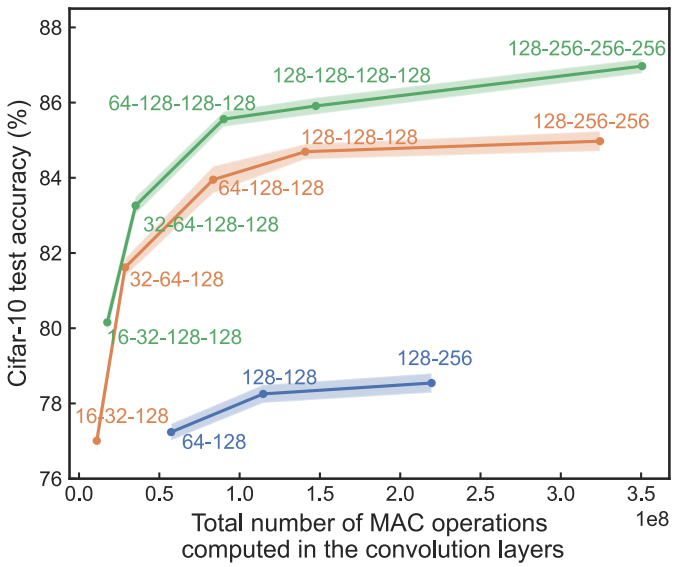


Figure 7.9: **Test accuracy of convolutional SPDNN models on CIFAR-10 classification.** The plot shows the mean test accuracy (data points) and the corresponding standard deviation (shaded region), calculated by averaging the results of 100 repeated inference runs. The number of output channels in the convolutional layers is indicated near each data point, with different colors denoting different numbers of convolutional layers. The linear layers following the convolutional layers have a consistent structure of $400 \to 10$, with SPD activation applied to the 400 neurons.

ers. The layout of the number of channels for each convolution layer is noted around each data point for each model. For example, "64–128" indicates that there are two convolution layers each with 64 and 128 output channels, respectively. These mean values (data points) and standard deviations (shaded area) of the test accuracies are obtained from 100 repeated inference, and the activations only involved a single shot of SPD measurement ($K = 1$) in all the neurons, including the convolutional and linear layers.

We further investigated the effects of multiple shots of SPD measurements per activation in the SPD activations in convolutional (K_{conv}) and linear (K_{lin})

layers, respectively. We chose to test a model with four convolutional layers of 128, 256, 256, and 256 output channels and varied the K_{lin} and K_{conv} to see the test accuracies. The results are summarized in Table 7.3.

	$K_{\text{lin}} = 1$	$K_{\text{lin}} = 2$	$K_{\text{lin}} = 3$	$K_{\text{lin}} = 5$	$K_{\text{lin}} = 10$	$K_{\text{lin}} \to \infty$
$K_{\text{conv}} = 1$	86.94 ± 0.23	87.11 ± 0.21	87.17 ± 0.21	87.23 ± 0.19	87.26 ± 0.20	87.28 ± 0.21
$K_{\text{conv}} = 2$	88.65 ± 0.18	88.77 ± 0.18	88.83 ± 0.18	88.88 ± 0.16	88.86 ± 0.18	88.90 ± 0.16
$K_{\text{conv}} = 3$	89.16 ± 0.15	89.26 ± 0.15	89.31 ± 0.16	89.33 ± 0.15	89.35 ± 0.14	89.39 ± 0.14
$K_{\text{conv}} = 5$	89.55 ± 0.14	89.68 ± 0.15	89.71 ± 0.14	89.74 ± 0.13	89.73 ± 0.13	89.77 ± 0.13
$K_{\text{conv}} = 10$	89.82 ± 0.12	89.91 ± 0.11	89.95 ± 0.10	90.00 ± 0.11	90.00 ± 0.12	90.02 ± 0.13
$K_{\text{conv}} \to \infty$	90.09 ± 0.09	90.20 ± 0.08	90.21 ± 0.07	90.25 ± 0.07	90.26 ± 0.05	90.31 ± 0.00

Table 7.3: **Test accuracy (%) of the convolutional SPDNN on CIFAR-10 with varying shots per activation K in the convolutional and linear layers.** The SPDNN model in this table consists of four convolutional layers with 128, 256, 256, and 256 output channels, respectively. The convolutional layers are followed by a linear layer with 400 neurons and an output layer with 10 neurons. The SPD activation function is applied to each of the 400 neurons in the first linear layer. K_{conv} represents the number of shots of SPD readouts per activation in the convolutional layers, while K_{lin} represents the shots per activation in the linear layer. The mean accuracy and standard deviation are calculated based on 100 repetitions of inferences using the CIFAR-10 test set of 10,000 images.

In these SPDNNs, the number of operations in the output layer is negligible compared to the entire models. In terms of the number of MAC operations (dot products, DPs), $N_{\text{MAC}}^{\text{out}} = 4000$ ($N_{\text{DP}}^{\text{out}} = 10$). The number of dot products, or the activation size, is directly related to the number of optical detections in ONN implementations. The output layer is the only layer that needs to be implemented with a "high SNR" (see Figure 7.1). The small portion of operations in this layer indicates the capability of low-SNR stochastic layers in a deeper model. This further suggests the potential to leverage stochastic physical systems with low SNRs to perform reliable neural-network inference.

Model	$N_{\text{MAC}}^{\text{total}}$	$N_{\text{MAC}}^{\text{out}}/N_{\text{MAC}}^{\text{total}}$	$N_{\text{DP}}^{\text{total}}$	$N_{\text{DP}}^{\text{out}}/N_{\text{DP}}^{\text{total}}$
C64–C128	5.73×10^7	6.60×10^{-5}	2.29×10^6	4.36×10^{-6}
C128–C128	1.15×10^8	3.40×10^{-5}	4.59×10^6	2.18×10^{-6}
C128–C256	2.20×10^8	1.80×10^{-5}	8.78×10^6	1.14×10^{-6}
C16–C32–C128	1.11×10^7	3.37×10^{-4}	4.42×10^5	2.26×10^{-5}
C32–C64–C128	2.87×10^7	1.36×10^{-4}	1.15×10^6	8.72×10^{-6}
C64–C128–C128	8.36×10^7	4.70×10^{-5}	3.34×10^6	2.99×10^{-6}
C128–C128–C128	1.41×10^8	2.80×10^{-5}	5.64×10^6	1.77×10^{-6}
C128–C256–C256	3.24×10^8	1.20×10^{-5}	1.30×10^7	7.71×10^{-7}
C16–C32–C128–C128	1.76×10^7	2.24×10^{-4}	7.05×10^5	1.42×10^{-5}
C32–C64–C128–C128	3.52×10^7	1.13×10^{-4}	1.41×10^6	7.10×10^{-6}
C64–C128–C128–C128	9.01×10^7	4.40×10^{-5}	3.60×10^6	2.77×10^{-6}
C128–C128–C128–C128	1.47×10^8	2.70×10^{-5}	5.90×10^6	1.70×10^{-6}
C128–C256–C256–C256	3.51×10^8	1.10×10^{-5}	1.40×10^7	7.13×10^{-7}

Table 7.4: **Number of operations in the convolutional SPDNN models.** The table displays the number of multiply–accumulate (MAC) operations and dot products (DPs) in the SPDNN models. $N_{\text{MAC}}^{\text{total}}$ ($N_{\text{DP}}^{\text{total}}$) represents the total number of MAC operations (dot products) in the entire SPDNN models, including all the convolutional layers and two linear layers. $N_{\text{MAC}}^{\text{out}}$ ($N_{\text{DP}}^{\text{out}}$) represents the number of MAC operations (dot products) in the output layer, which is the only layer implemented with a "high SNR". For the $400 \to 10$ output layer, $N_{\text{MAC}}^{\text{out}} = 4000$ and $N_{\text{DP}}^{\text{out}} = 10$. The portion of the high-SNR output layer's operations relative to the entire model's operations in terms of MAC operations (dot products) is presented in the third (fifth) column. "CN_{chan}" denotes an SPD convolutional layer with N_{chan} output channels.

7.3.3 Additional Classification Tasks

Apart from the benchmark MNIST and CIFAR-10 classification tasks, we added additional tests for KMNIST ("Kuzushiji MNIST") [245] and FashionMNIST [246] to further demonstrate the capabilities of our SPDNN models. To ensure fair and straightforward comparisons, we chose to test these tasks using the same structure we used for MNIST classification tasks, specifically the MLP with $784 \to 400 \to 10$. We compared the following four different models:

- A deterministic MLP with one hidden layer of 400 neurons, $784 \rightarrow 400 \rightarrow 10$. The weights are real-valued. The activation function is ReLU.
- A coherent MLP-SPDNN with one hidden layer of 400 neurons, $784 \rightarrow 400 \rightarrow 10$. The weights are real-valued. This model is the same as the one shown in Figure 7.6d ($N = 400$).
- An incoherent MLP-SPDNN with one hidden layer of 400 neurons, $784 \rightarrow 400 \rightarrow 10$. The weights are non-negative. This model is the same as the one shown in Figure 7.3b (MLP, 1 hidden layer).
- A linear classifier. The weights are real-valued.

Using a training process similar to that for the MNIST classification task (**??**), we optimized these SPDNN models with both incoherent and coherent optical setups. The blue dashed line represents the deterministic model with the same MLP architecture as the SPDNNs. The red dashed line represents the linear classifier. These lines set the upper and lower bounds of test accuracy, respectively, for each individual classification task.

By varying the number of shots per activation K in these SPDNN models, we observed trends consistent with those seen in the MNIST task (Figure 7.10), although some small differences exist among these tasks. For example, the test accuracy for coherent and incoherent MLP-SPDNNs tends to converge for the KMNIST classification task with increasing shots per activation K. However, this convergence is less pronounced for the other two tasks.

These additional results illustrate that our SPDNN models are effective across various tasks. They demonstrate that our highly stochastic SPDNN models can achieve reliable performance comparable to deterministic models.

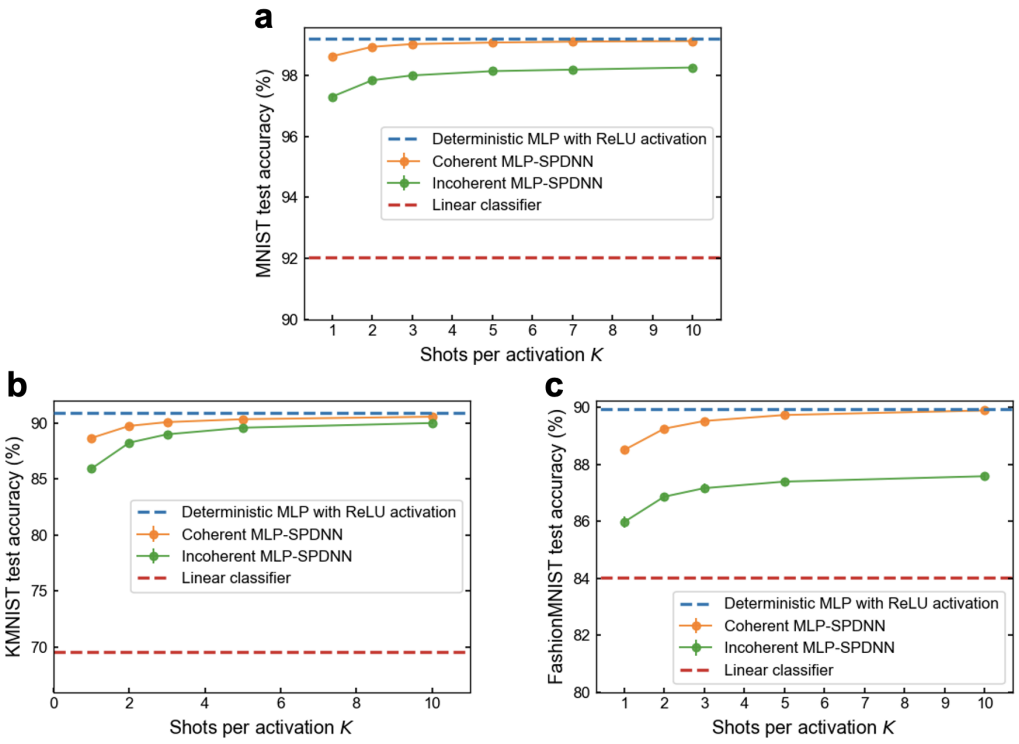


Figure 7.10: **Comparison of performance of different models on three image classification tasks.** The deterministic model and both coherent and incoherent SPDNNs share the MLP architecture of $784 \rightarrow 400 \rightarrow 10$. The x-axes represent the number of shots per activation K in the hidden layer of the corresponding SPDNN model. These models are trained for **a,** MNIST, **b,** KMNIST, and **c,** FashionMNIST classification tasks, respectively.

7.4 Experimental Implementation

In our experimental demonstrations, we based our SPDNN on a free-space optical matrix-vector multiplier (MVM) that we had previously constructed for high-SNR experiments [1], and replaced the detectors with SPDs so that we could operate it with ultra-low photon budgets (Section 7.4.1). The experiments we report were, in part, enabled by the availability of cameras comprising large arrays of pixels capable of detecting single photons with low noise [247].

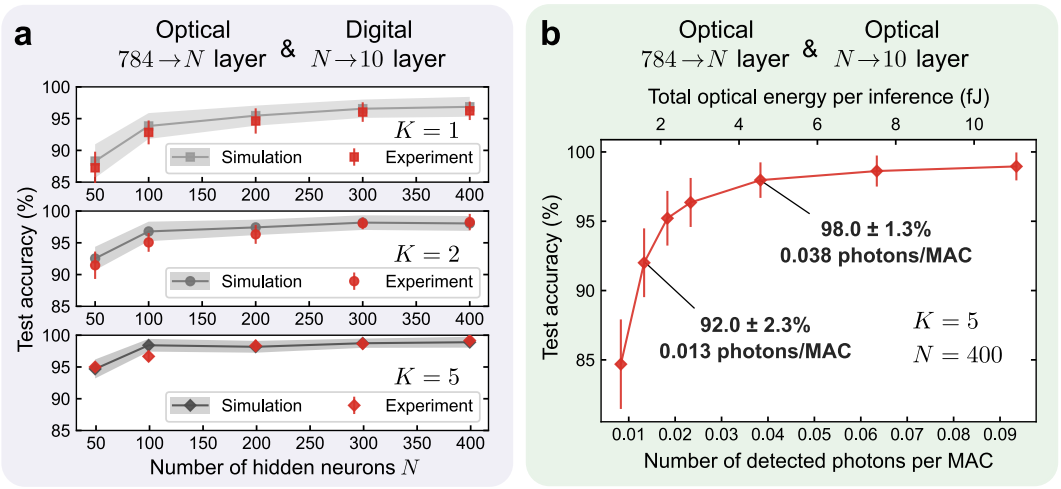


Figure 7.11: **Experimental performance of SPDNNs on MNIST handwritten-digit classification.** **a**, Experimental evaluation of the SPDNN, with the output layer performed with full numerical precision on a digital computer. Results are presented for both $K = 1$ (single-shot, i.e., no averaging; top), $K = 2$ (middle), and $K = 5$ (bottom) shots per activation. **b**, Experimental evaluation of the SPDNN, with both the hidden and the output layer executed using the optical experimental apparatus. The average number of detected photons used per inference in the hidden layer was kept fixed and the number used per inference in the output layer was varied. The number of detected photons per inference is reported both as an aggregate optical energy (top axis) and as a per-MAC quantity (bottom axis), which we obtained by dividing the number of photons per inference by the number of MACs performed in a single inference. The mean and standard deviation of the test accuracy were estimated used 100 repetitions of inference for each image in the test set.

We encoded neuron values in the intensity of incoherent light on an organic light-emitting diode (OLED) display; as a result, the weights and input vectors were constrained to be non-negative in a single operation. In the low-SNR layers, we imposed this non-negativity constraint during the training process. However, this is not a fundamental feature of SPDNNs—in the next section, we present simulations of coherent implementations that lift this restriction. A single-photon-detecting camera measured the photons transmitted through the optical MVM, producing the stochastic activations as electronic signals that were input to the following neural-network layer (Section 7.4.1).

In our first set of optical experiments, the hidden layer was realized optically and the output layer was realized *in silico* (Figure 7.11a): the output of the SPD measurements after the optical MVM was passed through a linear classifier executed with full numerical precision on a digital electronic computer. We tested using both $K = 1$ (no averaging) and $K = 2$ shots of averaging the stochastic binary activations in the hidden layer. In Figure 7.11a, we can see that the experimental results agree well with simulations that additionally modeled imperfections in our experimental optical-MVM setup, in contrast to the simulation results shown in Figure 7.3b that did not specifically consider these imperfections. The test accuracies were calculated using 100 test images, with inference for each image repeated 30 times. The hidden layer (the one computed optically in these experiments) used approximately 0.0008 detected photons per MAC, which is ≥ 6 orders of magnitude lower than is typical in ONN implementations [17, 18, 21, 46] and ≥ 3 orders of magnitude lower than the lowest photons-per-MAC numbers reported to date [1, 21].

We then performed experiments in which both the hidden layer and the out-

put layer were computed optically (Figure **??**d). In these experiments, we implemented a neural network with 400 hidden neurons and used 5 shots per inference ($N = 400$, $K = 5$). To execute the linear operations with real-valued weight elements on our incoherent setup, we divide the weight elements into positive and negative components. We perform the operations separately for each component and then obtain the final output values by subtracting the results of the negative weights from those of the positive weights. The total optical energy was varied by changing the number of photons used in the output layer; the number of photons used in the hidden layer was kept fixed (see details in Table 7.6 and Section 7.4.4).

The results show that even though the output layer was operated in the high-SNR regime (Figure 7.1b), the full inference computation achieved high accuracy yet used only a few femtojoules of optical energy in total (equivalent to a few thousand photons). By dividing the optical energy by the number of MACs performed in a single inference, we can infer the per-MAC optical energy efficiency achieved: with an average detected optical energy per MAC of approximately 0.005 attojoules (or 0.014 attojoules) at a photon wavelength of 532 nm, equivalent to 0.013 photons (or 0.038 photons), the mean and standard deviation of test accuracy achieved 92.0 ± 2.3% (or 98.0 ± 1.3%). The optical energy per MAC serves as a metric of the overall SNR at which the system operates, allowing for direct comparison with previous works. The end-to-end system energy consumption advantage is beyond the scope of this paper.

We can also compare our results with what has been published previously. Our experiments, with $N = 50$ hidden neurons and $K = 5$ shots of SPD measurements per activation (see Figure 7.24) achieved a test accuracy of 90.6% on

MNIST handwritten-digit recognition while using only an average of 1390 detected photons per inference (corresponding to ~0.5 fJ of detected optical energy per inference). This represents a >40× reduction in the number of photons per inference to achieve >90% accuracy on this task versus the previous state-of-the-art [1, 21].

Despite these already very low numbers, most photons per inference in these implementations are consumed in the high-SNR output layer, and there is still potential for further reduction with a more optimized experimental setup (Section 7.4.4). As we discussed in Section 7.3.2 (Table 7.4), as models scale up, the impact of a single high-SNR output layer becomes negligible.

In the following sections, we detail the process to obtain these results on the optical MVM reported in Chapter 3.

7.4.1 Calibration of the Setup

Single-photon detection by a scientific CMOS camera

Single-photon detection is core to implementing an SPDNN. In our experiment, we use a scientific CMOS camera, the Hamamatsu ORCA-Quest qCMOS Camera, to realize the function of single-photon detectors [247]. CMOS cameras usually cannot detect single photons due to the relatively high readout noise compared to the signals induced by individual photons. The ORCA-Quest qCMOS camera, however, has well-controlled readout noise as low as 0.3 equivalent photoelectrons. This makes viewing the individual spikes of photon response possible on the output of the camera. An example of the distribution of pixel

values from the camera is shown in Figure 7.12a. These pixel values are from a sequence of frames collected with some intensity of input light. The output pixel values have a digital bias of ≈ 200, and the analog gain is ~ 7.94 pixel values per photoelectron. We can see the individual spikes corresponding to different numbers of detected photons, with the first peak referring to no detected photons. Due to readout noise of the camera, we can still see a near-Gaussian distribution around the peak value of each detected photon number. To do single-photon detection, a threshold can be set to determine if there is a photon (or more photons) detected. If the pixel value is larger than the threshold, we record a click; otherwise, there is no click. In this way, although the camera has already completed analog-to-digital conversion (ADC) before thresholding, the qCMOS camera can still emulate the function of a single-photon detector.

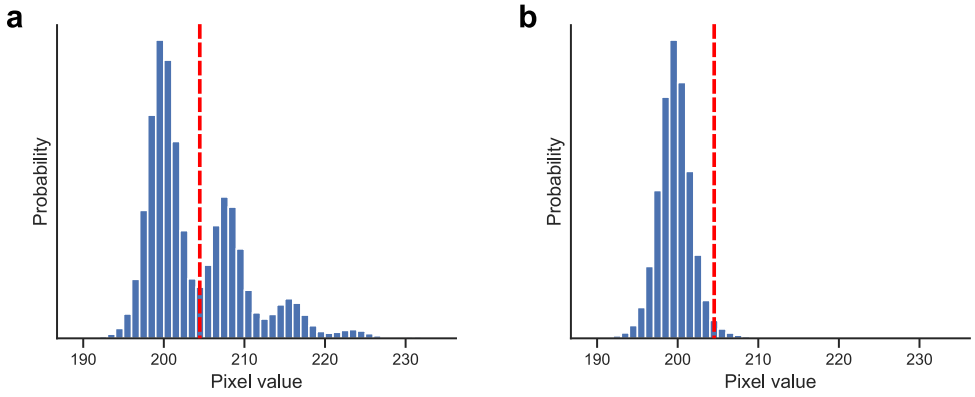


Figure 7.12: **Pixel value distributions of the ORCA-Quest qCMOS camera.** **a,** An example of pixel value distribution in collected frames with input signal light of a few photons. The red dashed line indicates the threshold to tell if there is a photon click. We can see clearly that the discrete numbers of photons can be resolved in the individual peaks. **b,** Pixel value distribution of the dark frames without input light.

This camera has an overall quantum efficiency of $\sim$ 86% at our working wavelength of 532 nm and a dark count rate of 0.006 photoelectrons per sec-

ond per pixel at the working temperature of -35ºC with air cooling. Due to the readout noise, the thresholding process may add additional errors because of the overlap of the signal peaks. Figure 7.12b shows the pixel value distribution of dark frames without input signals. We can see that using the same threshold, there is a small tail of the distribution on the right side of the threshold. This small portion of the pixel values from dark frames would trigger a photon click as well, which further adds to the dark count rate. Similarly, the output pixel values from detected photons also have a small probability to fall in the "no click" region, which makes the effective photon detection efficiency a bit lower. In our experiment, we calibrated the qCMOS camera in the single-photon detection mode and found that the effective dark count rate of around 0.01 photoelectrons per second per pixel, and the effective photon detection efficiency to be 68%, on average. Note that there are also variations among different pixels. In the experiment implementation, only one single pixel is used for each vector-vector dot product. We will see that the photon detection efficiency and dark counts do not significantly influence the results, as discussed in Section 7.5.1.

Validation of the Optical Vector-Vector Multiplications

The major part of computation in the ONN implementation is the linear operations. The accuracy of matrix-vector multiplication is essential in a successful ONN inference. In this section, we calibrate the accuracy of our optical MVM. We use the setup with either single-photon detection or conventional intensity measurement that involves a much higher intensity. Focusing the lights to one pixel of ~ 5 μm is challenging, which reduces the dot product precision slightly. However, as we will see in Section 7.5.1, the SPDNN models are very robust to

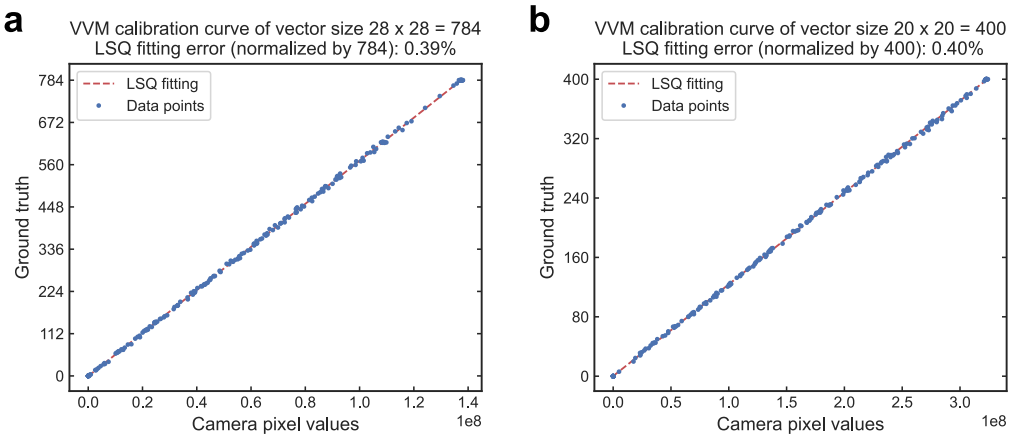


Figure 7.13: **Calibration curves of vector-vector multiplication (VVM) precision in the setup.** Panel **a** (**b**) shows the VVM calibration of the vector size of 784 (400), which were used for the hidden (output) layer in the optical implementation of the SPDNN model with a structure of $784 \rightarrow 400 \rightarrow 10$. The calibration curves (red dashed line) were obtained using the least-squares regression method and the data points are collected with a long exposure time to eliminate the photon noise.

this amount of errors.

To generate a test dataset representative of general dot products, we randomly generated vector pairs $\vec{x}$ and $\vec{w}$ based on natural scene images from the STL10 dataset. Each vector was generated from a single color channel of one or more images patched together, depending on the target vector size (each image of size $L \times L$ contributes $N = L^2$ elements to the vector). We chose natural images since they are more representative of the inputs in image classification with globally inhomogeneous and locally smooth features. To adjust the sparsity of the vectors, different thresholds were applied to the image pixel values such that the dot product results cover a wider range of possible values. This was achieved by shifting the original pixel values (float point numbers normalized to the range 0-1) in the entire image up or down by a certain amount, unless the

value was already saturated at 1 (maximum) or 0 (dark). For example, a shift of -1 would make the whole image dark. A shift of +0.2 would make all the pixel values that were originally larger than 0.8 saturated, and would increase all other pixel values by 0.2. This method allowed us to tune the overall intensity of the modulated images without losing the randomness of the distribution.

Calibration curves of vector-vector dot product results are shown in Figure 7.13. The results are averaged over a large number of repetitions to get rid of the photon noise to see the systematic errors in the optical MVM. The vectors are randomly generated to cover the full range of the light intensity from the minimum to maximum transmission, as discussed in [1]. The vector size is 28×28, which is equivalent to the size of the first layer in MNIST classification.

Validation of the SPD Activation Function

To validate the SPD activation function in the SPDNN implementation, we need to consider not only the precision of the linear operations, but also the non-linear activation function. As the incident light onto the qCMOS camera is attenuated to just a few photons, the photon noise becomes significant and the measurement less accurate. To address this, we first measure a higher light intensity with long exposure times and estimate the exact light intensity with a shorter exposure time using the ratio of exposure times. We then use the shorter exposure time to perform single-photon detection to output a photon click (value 1) or no photon click (value 0). The probability of a photon click is estimated by averaging over a large number of repetitions. The intensity is tuned by adjusting both the exposure time and neutral density (ND) filters that attenuated the light. The expected theoretical curve is also plotted for compari-

son. The results are shown in Figure 7.14.

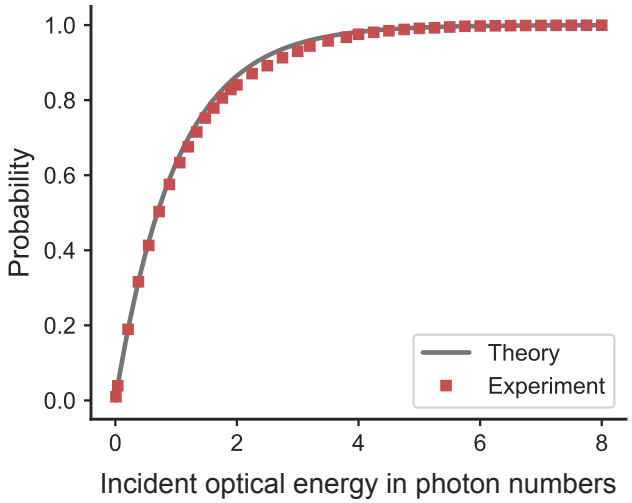


Figure 7.14: **Validation of the SPD activation function.** The theory curve is the expected function of $f(\lambda) = 1 - e^{-\lambda}$ with the intensity λ in photon numbers. The experiment data was taken by the Hamamatsu ORCA-Quest qCMOS camera.

7.4.2 Adaptation to Experimental Limitations

The implementation of SPDNNs on an optical MVM can be challenged by experimental restrictions that affect the precision of the network inference. Some of these limitations include the non-negative encoding resulting from the use of an incoherent light source and limitations in the precision of the setup. In this section, we describe how these limitations can be addressed to successfully implement SPDNNs on our setup. As discussed in Section 7.4.1, our incoherent optical MVM has systematic errors in the dot-product results, even in the absence of photon noise. Additionally, the SLM used in the system has a finite

extinction ratio of approximately 50 (Figure **??**b). These limitations present a significant challenge in the implementation of the SPDNNs because, in the models, both the input vectors and weights have many small values close to 0. This is problematic because, within the full range of 0 to 1, having a minimum value of 0.02 instead of 0 has a non-trivial effect on the accuracy of the dot product calculation. These small values are accumulated over many elements, leading to a relatively large value, compared to the final dot product result. As a result, the performance of the SPDNNs is severely impacted by these limitations.

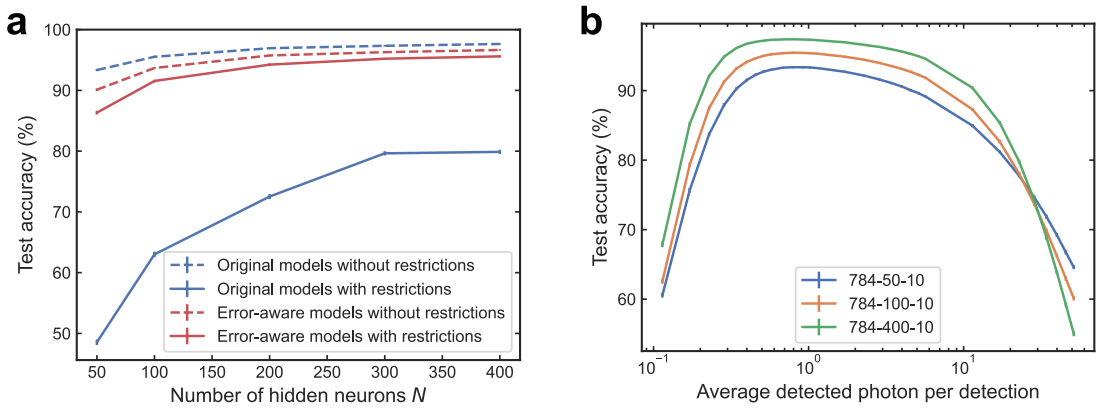


Figure 7.15: **Simulation of SPDNN performance with different experimental settings. a,** MNIST test accuracy of SPDNN models under experimental restrictions. The models have a structure of $784 \rightarrow 400 \rightarrow 10$ ($N = 400$, $K = 1$). **b,** MNIST test accuracy as a function of the input light intensity. The intensity was varied by adjusting the range of input values using a constant factor. Both panels show results obtained with an incoherent setup and a single shot of SPD readout per activation ($K = 1$).

Figure 7.15a demonstrates the results of implementing the neural network models using the real LUTs from our setup. The test accuracy significantly drops, making the experimental implementation a failure. To address this issue, we used error-aware training techniques (as discussed in [248]) to train our

models with an understanding of these experimental restrictions. During the error-aware training process, the real LUTs were used in the implementation of the models. The results of this error-aware training are shown in the red curves in Figure 7.15a. It can be seen that, with error-aware training, the SPDNN models are highly robust to changes in input range, especially with a relatively large number of hidden neurons.

Conventional ONN inferences can operate effectively at various light intensity levels, as long as the intensity is sufficiently high to suppress photon noise and maximize detection precision. These systems can, in principle, integrate arbitrarily high light intensities to enhance detection precision. However, in the optical implementation of SPDNN inferences, the SPD activation function relies on the precise number of photons detected. As a result, controlling the operating intensity in the setup becomes crucial to ensure accurate quantization of the detected optical energy. Calibrating the intensity to the appropriate level for SPD activation function presents a challenge, especially considering the inherent significant noise in intensity measurements at low photon counts.

Despite these challenges, our simulation results demonstrate the robust performance of SPDNNs even with slight variations in the input intensities. We systematically varied the input intensity across a range from 0.1 to 100 times the original expected intensity that was used during training. Figure 7.15b illustrates that the model's performance remains stable within a wide range of intensities. The test accuracy remains nearly consistent, even when the input energy deviates significantly from the original training intensity. This observed stability highlights the resilience of SPDNNs to variations in input intensity levels. It further suggests that these SPDNN models can be successfully implemented

with lower photon budgets, which is promising for practical applications where minimizing optical energy usage is desirable.

7.4.3 Optical Implementation of the SPD Activations

In this section, we demonstrate the experimental implementation of SPD activation functions in SPDNN inferences. The weight matrix of the first layer $W^{(1)}$ in the SPDNNs was displayed on the OLED screen, with each element encoded as the intensity of a corresponding pixel. The models used in experiments are trained to account for the limitations of the experimental setup, as discussed in Section 7.4.2. In Figure 7.16, the weights of the model with $N = 100$ hidden neurons are depicted. The two-dimensional arrangement of each weight vector block mirrors the visual representation on the OLED display, which matches each pixel of the MNIST input images. The MNIST test images were displayed on the SLM, where the pixels were aligned with those on the OLED display. The transmission of each SLM pixel is determined by the values of the corresponding pixels in the input image. After passing through the SLM, the modulated intensity of each pixel is then combined through the imaging system to perform an optical fan-in process, and the dot product result is obtained by detecting the accumulated optical energy.

As described in Section 7.4.1, the qCMOS camera serves as a single-photon detector (SPD). The modulated light is recorded by the qCMOS camera, and the SPD activation function is applied during this single-photon detection process. A single pixel on the qCMOS camera is used and emulates a single-photon detector by applying a threshold on the output pixel values. As shown in Figure

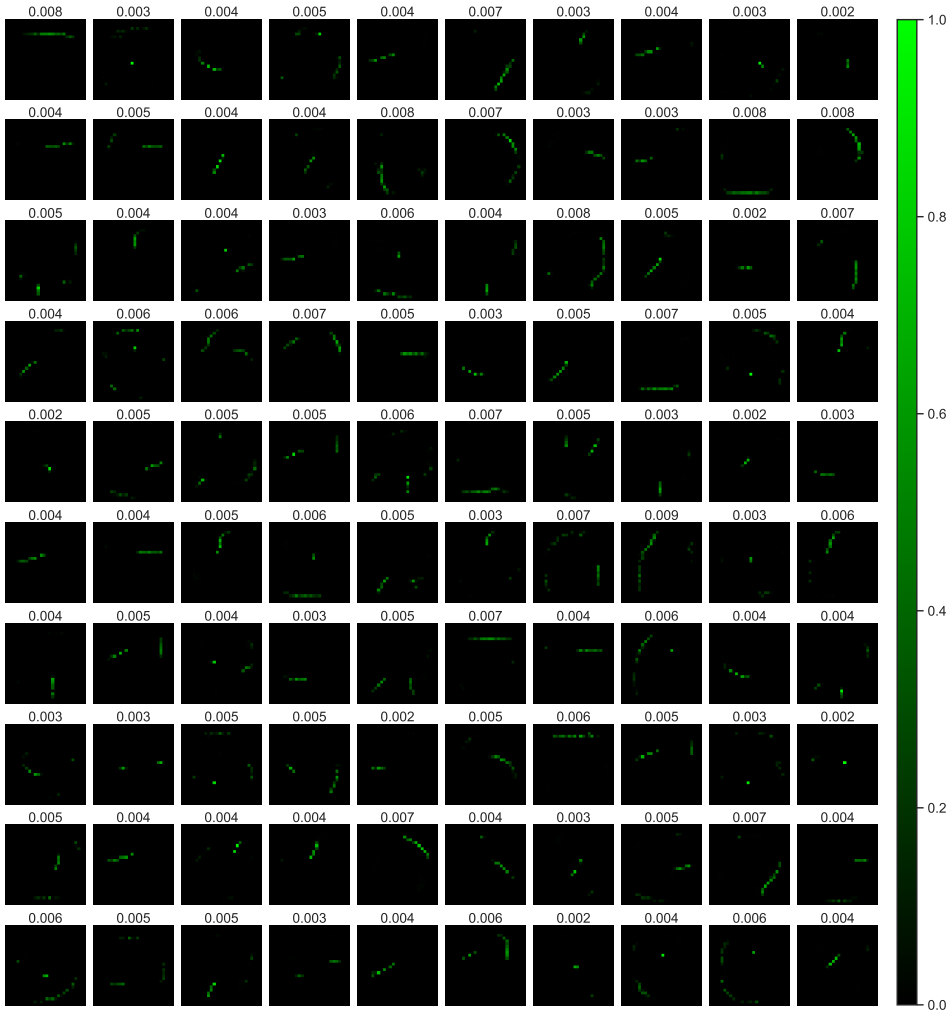


Figure 7.16: **An example of weight values to be displayed on the OLED display during the experiment.** This model has $N = 100$ hidden neurons, and we display the weight matrix $W^{(1)}$ of the first layer (with dimensions 100×784). Each block represents a row vector in $W^{(1)}$ containing 784 elements. These column vectors are rearranged to form a 2D block with dimensions 28×28, matching the original shape of the MNIST input images. The 100 rows in $W^{(1)}$, corresponding to the 100 hidden neurons in the neural network, are arranged in a 10×10 grid to be visualized. The weight values have been normalized to a range of 0 to 1 to fit the intensity range of the OLED display. The average value of each block is indicated at the top. The color map used in the plot has been selected to emulate the actual color on the OLED display, as only green pixels are utilized ($\sim$ 532 nm), thereby presenting what could be observed on the OLED display in the experimental setup.

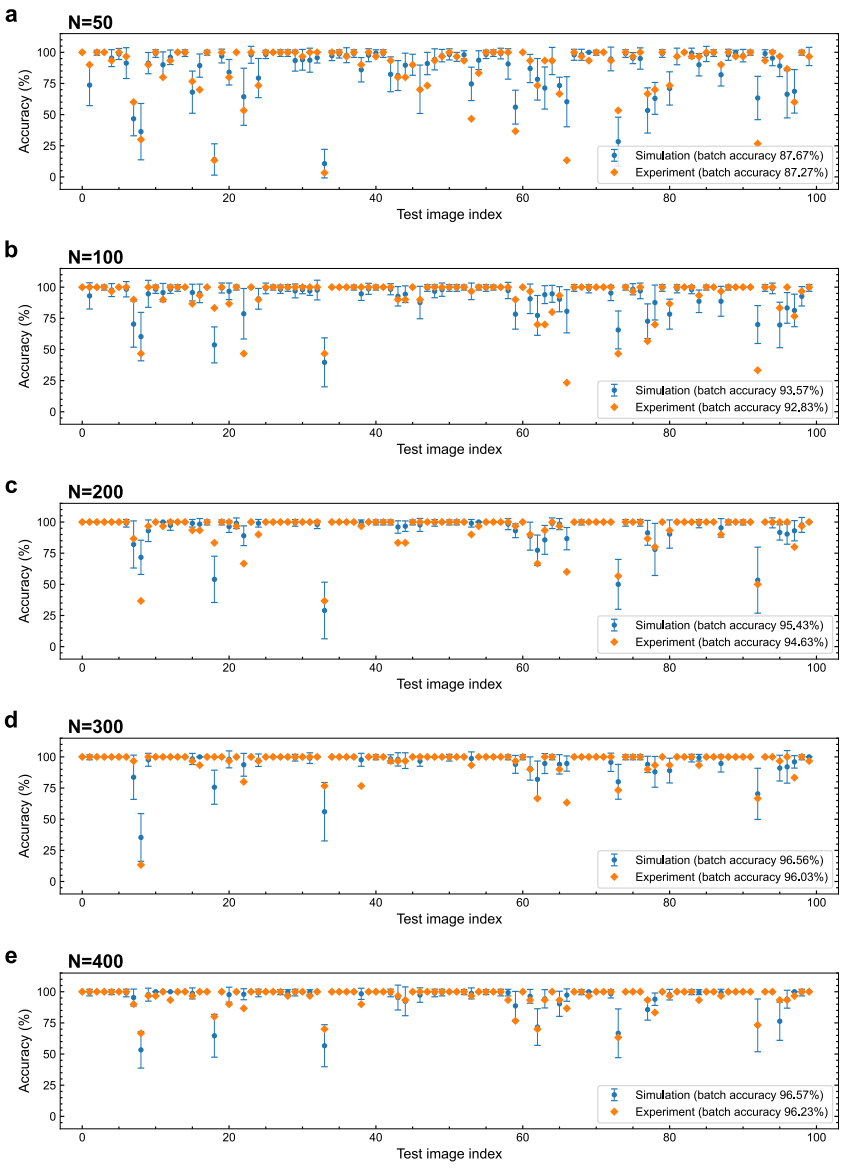


Figure 7.17: **Test accuracy of individual images with 1 SPD measurement per activation ($K = 1$).** The MLP-SPDNN models with a varying number of hidden neurons N from 50 to 400 (panels **a**–**e**) were evaluated using a single shot per SPD activation ($K = 1$). Test accuracy was evaluated for each individual image, and each data point represents the average accuracy obtained from 30 inferences. We have not plotted error bars for experimental data, but the variance is directly related to the data shown: since the outcome of each inference is a Bernoulli random variable with p, the variance is $p(1 - p)$. The accuracy values in the legends are averaged over all test images. The simulation results used the same SPDNN models as in the experiment and considered the experimental restrictions. The error bars were calculated by repetitions of the whole process.

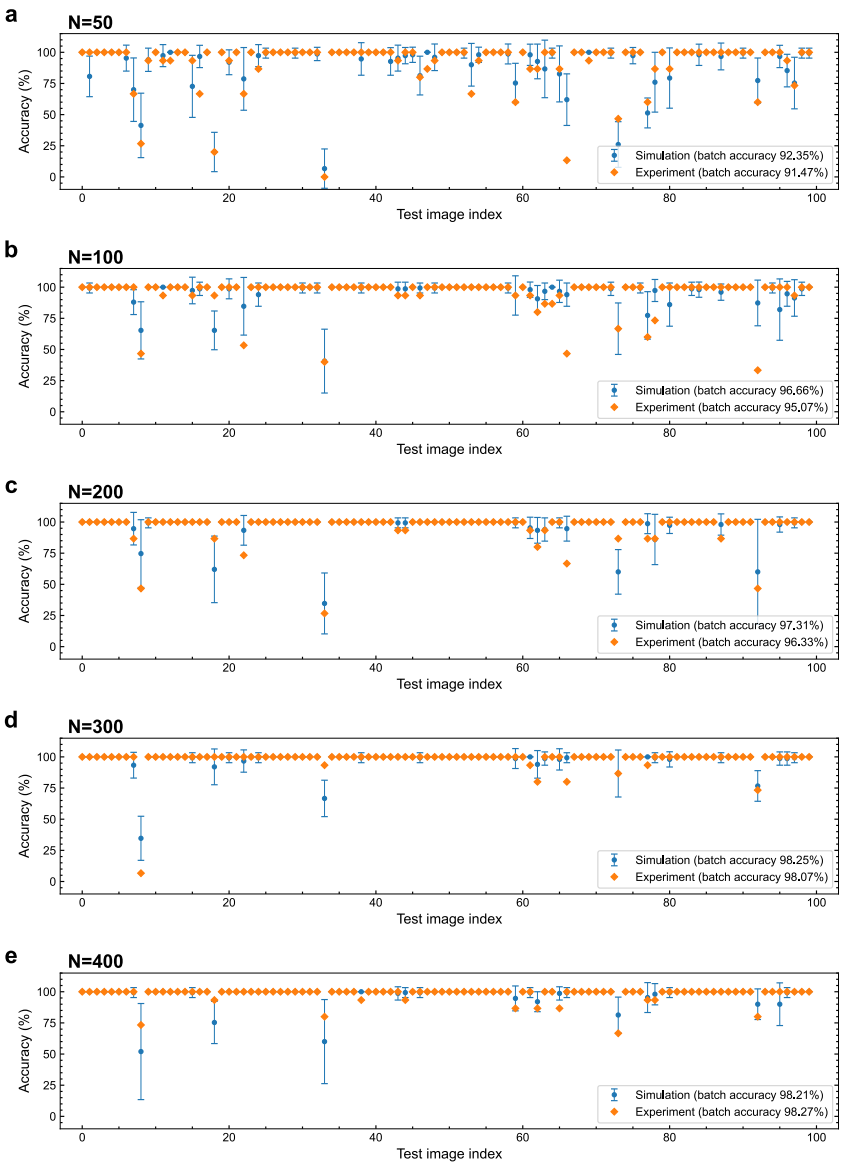


Figure 7.18: **Test accuracy of individual images with 2 SPD measurements per activation ($K = 2$).** The MLP-SPDNN models with a varying number of hidden neurons N from 50 to 400 (panels **a**–**e**) were evaluated using 2 shots per SPD activation ($K = 2$). Test accuracy was evaluated for each individual image, and each data point represents the average accuracy obtained from 15 inferences. We have not plotted error bars for experimental data, but the variance is directly related to the data shown: since the outcome of each inference is a Bernoulli random variable with p, the variance is $p(1-p)$. The accuracy values in the legends are averaged over all test images. The simulation results used the same SPDNN models as in the experiment and considered the experimental restrictions. The error bars were calculated by repetitions of the whole process.

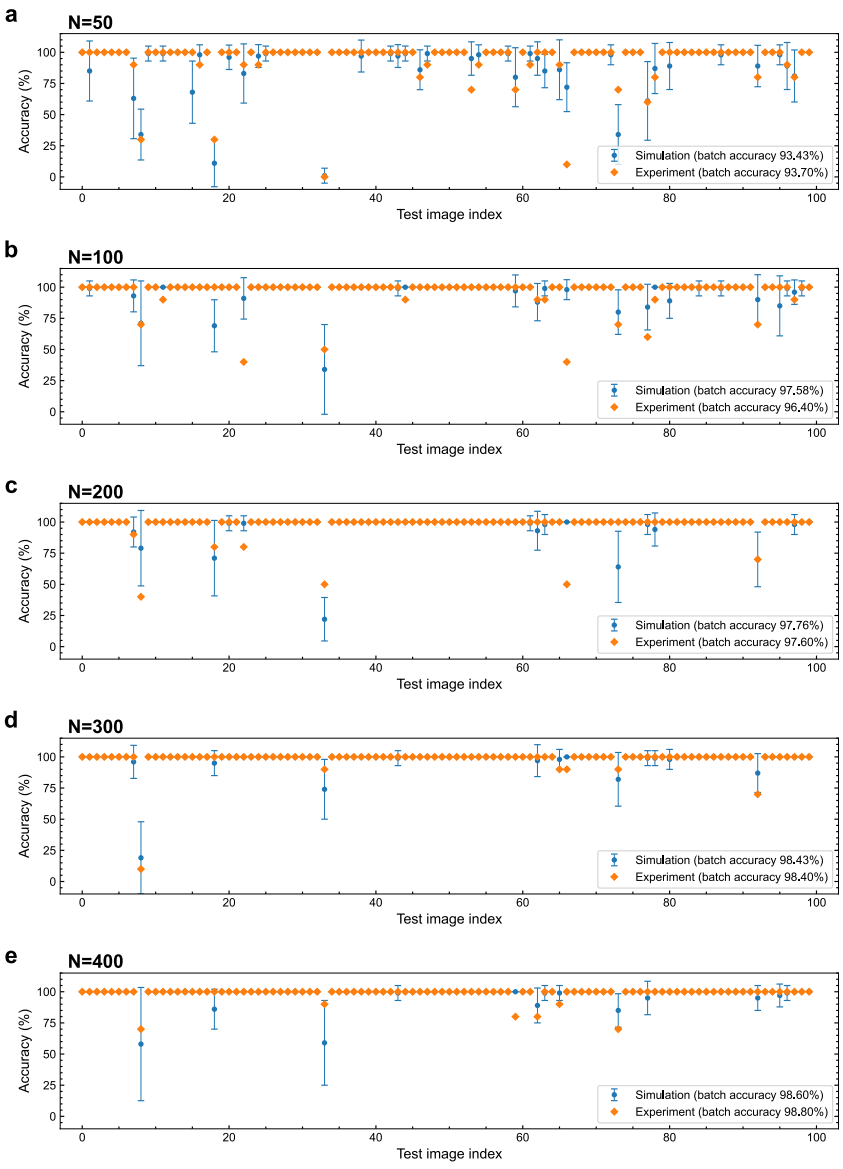


Figure 7.19: **Test accuracy of individual images with 3 SPD measurements per activation ($K = 3$).** The MLP-SPDNN models with a varying number of hidden neurons N from 50 to 400 (panels **a**–**e**) were evaluated using 3 shots per SPD activation ($K = 3$). Test accuracy was evaluated for each individual image, and each data point represents the average accuracy obtained from 10 inferences. We have not plotted error bars for experimental data, but the variance is directly related to the data shown: since the outcome of each inference is a Bernoulli random variable with p, the variance is $p(1-p)$. The accuracy values in the legends are averaged over all test images. The simulation results used the same SPDNN models as in the experiment and considered the experimental restrictions. The error bars were calculated by repetitions of the whole process.

7.12, the output signal is primarily discrete in individual photon numbers, with only slight mixing due to readout noise from the electronic apparatus. Robustness to the false clicks due to the mixing will be discussed in Section 7.5.1. To control the input light intensity, either the exposure time is varied or light attenuation can be achieved through the use of neutral density filters, or a combination of both, depending on realistic experimental settings. A typical exposure time value is ~ 1 ms for one shot of SPD measurement.

Due to the inherent stochastic nature of SPDNN models, the output from different repetitions of the same test image and weight matrix varies. To assess the performance of the model, it is necessary to repeat the inference many times to represent the distribution of the output. This will give us a better understanding of the model's behavior and its accuracy. In the experimental setup, multiple repetitions of the inference were obtained by capturing multiple frames with the camera. Each repetition is identified by a unique frame index number. Note that this repetition includes the entire inference process. Each repetition of inference for a given input image involves $N \times K$ SPD measurements, with K shots of SPD measurements for each of the N neuron activations in the hidden layer.

In our experiments, we collected a set of 30 frames for each test image and weight matrix combination. Each frame provides us with a measurement of the stochastic binary output values produced by the SPD activation function, effectively serving as one-shot activations for each hidden neuron in the model. The activations for different frames, distinguished by different frame indices, are considered as distinct repetitions of the inference process.

As explained in Algorithm 2, we can improve the precision of the inference by employing multiple shots per activation. With K shots, we average the K

binary values to obtain the actual activation value during that inference. In our experiment, every K frames are averaged to obtain the activation value during a K-shot inference. As each frame collected during the experiment is independent and identically distributed, the specific sequence or arrangement of the frame indices has no impact on the resulting outcome. Subsequently, the activation values were used as inputs to the output linear layer, performed digitally with full precision, to calculate the final output and make predictions for the label of the test image. The test accuracy results can be found in Table 7.5.

Our experimental results show that the collected SPD activations are capable of producing similar test accuracy results to those obtained from simulation. However, simply having similar accuracy values may not be a sufficient indicator of a faithful implementation. To validate this further, we compare the prediction accuracy of individual test images between the experimental and simulated results. The results obtained from 1-shot inferences $K = 1$ are presented in Figure 7.17, where we use each frame as a separate repetition of the inference process, with a total of 30 repetitions performed. For each repetition, if the prediction made was accurate, it was recorded as a 1; otherwise, it was recorded as a 0. To visualize the distribution of the output accuracy, we then calculated and plotted the mean values and standard deviations of the test accuracy based on the 30 repetitions.

We conducted simulations of the same inference process on a digital computer using the same models and input images. To ensure a closer simulation to reality, we also incorporated realistic experimental restrictions, such as the limited extinguish value of the SLM, the dynamic range and precision of the LUT in both the SLM and the OLED display, and the systematic errors in the optical

MVM.

Similarly, we examined the results of the inferences with $K = 2$ ($K = 3$) shots per activation, which are illustrated in Figure 7.18 (7.19). In this setup, we combine every 2 (3) frames to be averaged to compute a neuron activation, and we repeated this process 15 (10) times to obtain the final results.

By comparing the simulation results with the experimental results obtained from the collected SPD activation values, we aimed to validate the performance of the latter. The comparison revealed that, for the majority of input images, the predictions are highly resilient to the inherent stochasticity in the model. Interestingly, the results are not as unpredictable as one might expect, as a closer examination shows that most of the errors stem from a limited number of specific input images (see Figures 7.17–7.19).

Model	$K = 1$	$K = 2$	$K = 3$	$K = 5$	$K = 10$
784 – 50 – 10	87.3 ± 2.5%	91.5 ± 2.2%	93.7 ± 1.8%	95.0 ± 0.0%	95.0 ± 0.8%
784 – 100 – 10	92.8 ± 1.9%	95.1 ± 1.5%	96.4 ± 1.2%	96.7 ± 0.7%	97.7 ± 0.5%
784 – 200 – 10	94.6 ± 2.0%	96.3 ± 1.5%	97.6 ± 1.3%	98.3 ± 0.9%	98.3 ± 0.9%
784 – 300 – 10	96.0 ± 1.5%	98.1 ± 0.9%	98.4 ± 1.1%	98.7 ± 0.5%	98.7 ± 0.5%
784 – 400 – 10	96.2 ± 1.5%	98.3 ± 1.3%	98.8 ± 0.9%	99.2 ± 0.9%	99.7 ± 0.5%

Table 7.5: **Test accuracy of the experimental SPD activations with different shots per activation K.** The results are based on 30 repetitions of one-shot binary SPD activations collected for each model structure. The test accuracy was calculated by averaging K shots per activation, and the last layer was performed with full precision. The table displays the mean and standard deviation of the test accuracy obtained from $30/K$ repetitions of inference.

The close correspondence between the experimental and simulated results for these specific "problematic" input images further validates the reliability of our experimental implementation. Although the experimental results are slightly inferior to the simulation results, the distribution of accuracy per input image is highly comparable. In particular, input images that exhibit high sensi-

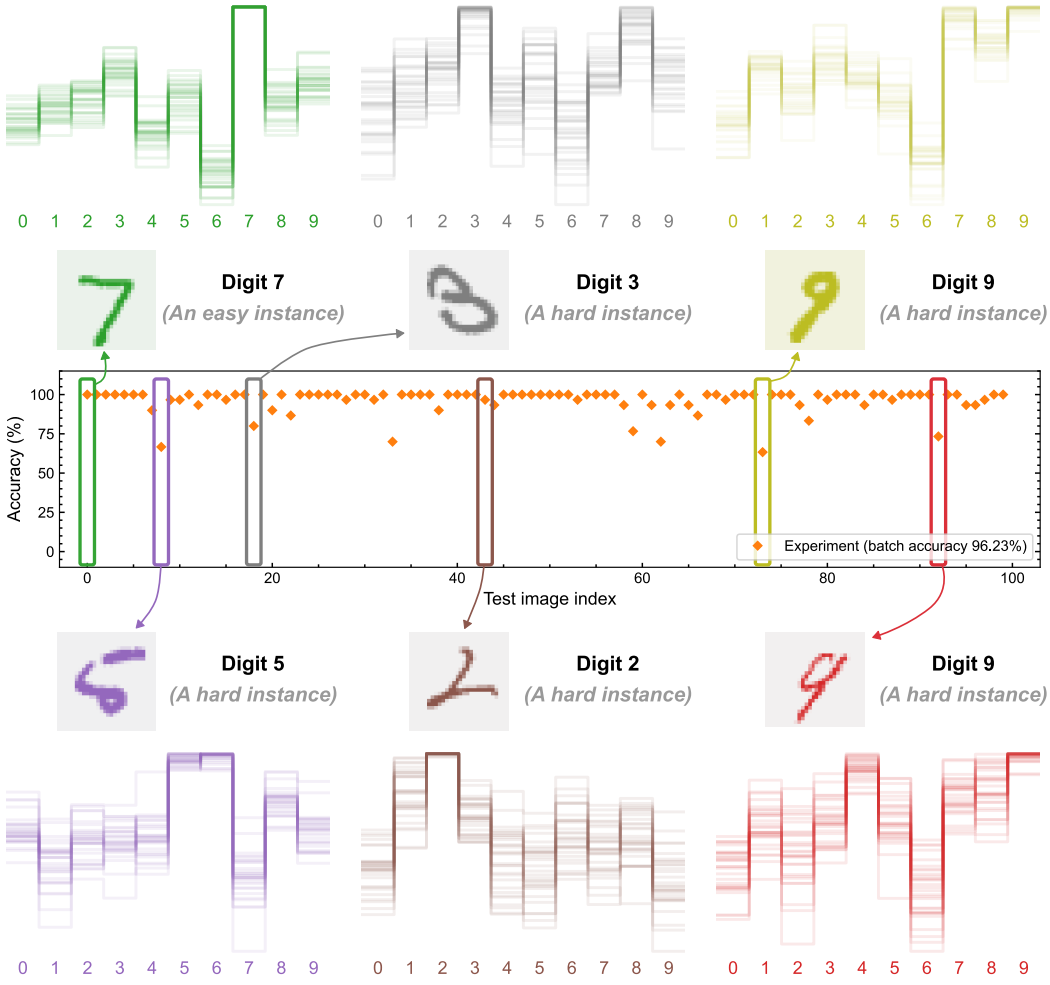


Figure 7.20: **Visualization of the output vectors of the SPDNN with given input images.** In this figure, the SPDNN model has $N = 400$ hidden neurons and $K = 1$ shot of SPD measurement per activation (Figure 7.17e). The prediction of a single inference of a particular test image is stochastic and is either correct or incorrect. For each test image, we performed 30 inferences for each test image and report the average accuracy. We have not plotted error bars, but the variance is directly related to the data shown: since the outcome of each inference is a Bernoulli random variable with p, the variance is $p(1-p)$. The output vectors from the 30 repetitions of inference with the fixed corresponding image are plotted together, with a 10% transparency on the curves to show the density. These output vectors are computed from the experimentally collected SPD activations by performing the output layer digitally on a computer (Table 7.5).

tivity to the model's stochasticity tend to result in larger deviations in the experimental results, while input images that are robust to the model's stochasticity exhibit high accuracy both in simulations and in experiments. These results provide strong evidence of the reliability of the experimental implementation and demonstrate the robustness and noise resilience of SPDNN implementations.

To further understand the characteristics of the stochastic neural-network inference, we examined the output vectors of each input test image. As depicted in Figure 7.20, the 30 output vectors from different repetitions of each input image are plotted together to demonstrate the stochasticity in the neural network. These output vectors were computed by the experimentally measured SPD activations and digitally implemented output layer, with $N = 400$ hidden neurons and $K = 1$ shot of SPD measurement per activation (Figure 7.17e). No additional operations were performed after the linear operation of the output layer (see Algorithm 2).

Each of the 10 values in the output vector corresponds to the classes in MNIST digit classification, ranging from 0 to 9, as indicated at the bottom. The curves of the 30 output vectors were plotted with 10% transparency to show the distribution density.

As shown in Figures 7.17–7.19, most of the test images have very high accuracy and are predicted correctly by SPDNN with high certainty, such as image 0 of digit "7" (depicted in the upper left in Figure 7.20). Despite the stochastic distribution of the output values among the 30 repetitions, the value of class "7" remains consistently higher than the other elements, resulting in a 100% test accuracy for this image (Figure 7.1a).

We also examined these "problematic" images, such as image 8 of digit "5" (lower left), which is predicted to be digit "6" nearly half of the chance. This misclassification is not surprising to human observers, as the image shares features with both digits "5" and "6". Interestingly, the output values for class 8 in this case are relatively high but not the highest, which also aligns with human intuition.

Similar phenomena can be found for the other "problematic" images as well, indicating that the model has indeed learned meaningful features from the dataset. These findings solidify the fact that stochastic neural networks can perform reliable deterministic classification tasks, and the inherent stochasticity in the model does not compromise its ability to make accurate predictions.

7.4.4 Full-Optical Implementation

In this section, we showcase the full-optical implementation of SPDNN models by demonstrating the implementation of the last linear layer optically as well, using the SPD activation values obtained from the inference of the first layer. This provides a comprehensive illustration of the feasibility of optical implementation for the entire network. It is important to note that, in conventional binarized neural networks, the last layer is usually implemented using full precision, as demonstrated in previous studies such as [225, 249, 250, 243]. Our results demonstrate that SPDNNs can be implemented entirely using optics with remarkably low energy requirements. This capability holds promise for further advancements, especially with the integration of coherent optical computing platforms, which will be discussed later.

Similar to the first layer, we use the same setup to perform the optical matrix-vector multiplication. The difference is that now we do not need to perform single-photon detection that has to control the light intensity at a few photons per detection. In fact, the inference of the last linear layer can be implemented just as the conventional ONNs, where we accumulate a sufficiently high number of photons to reach a high SNR for each detection. The collected SPD activation values, as described in Section 7.4.3, are used as inputs to the last linear layer. In the experimental implementation, we choose the data from the model with $N = 400$ hidden neurons and $K = 5$ shots per activation. For the 30 frames of one-shot binary SPD activations, every 5 frames of them are averaged to obtain the 6 independent repetitions of the inference. The input activation values to be displayed on the SLM are shown in Figure 7.21. The possible values for the 5-shot activations are 0, 0.2, 0.4, 0.6, 0.8, and 1. If the linear operation was performed in full precision on a computer, the mean test accuracy would be approximately 99.1%. To perform the linear operation with real-valued weight elements on our incoherent setup, we divide the weight elements into positive and negative parts. We perform the operation separately for each part and finally obtain the output value by subtracting the results with negative weights from those with positive weights. The two sets of weights to be projected onto the OLED display are shown in Figure 7.22, where the ten blocks of weights correspond to the ten output nodes. This approach at least doubles the photon budget required for the last layer and has the potential to be optimized for greater energy efficiency. However, even with these non-optimized settings, our results demonstrate that the optical energy budget is already several orders of magnitude lower than the state-of-the-art ONN implementations.

In the implementation, we adjust the exposure time of the camera to control

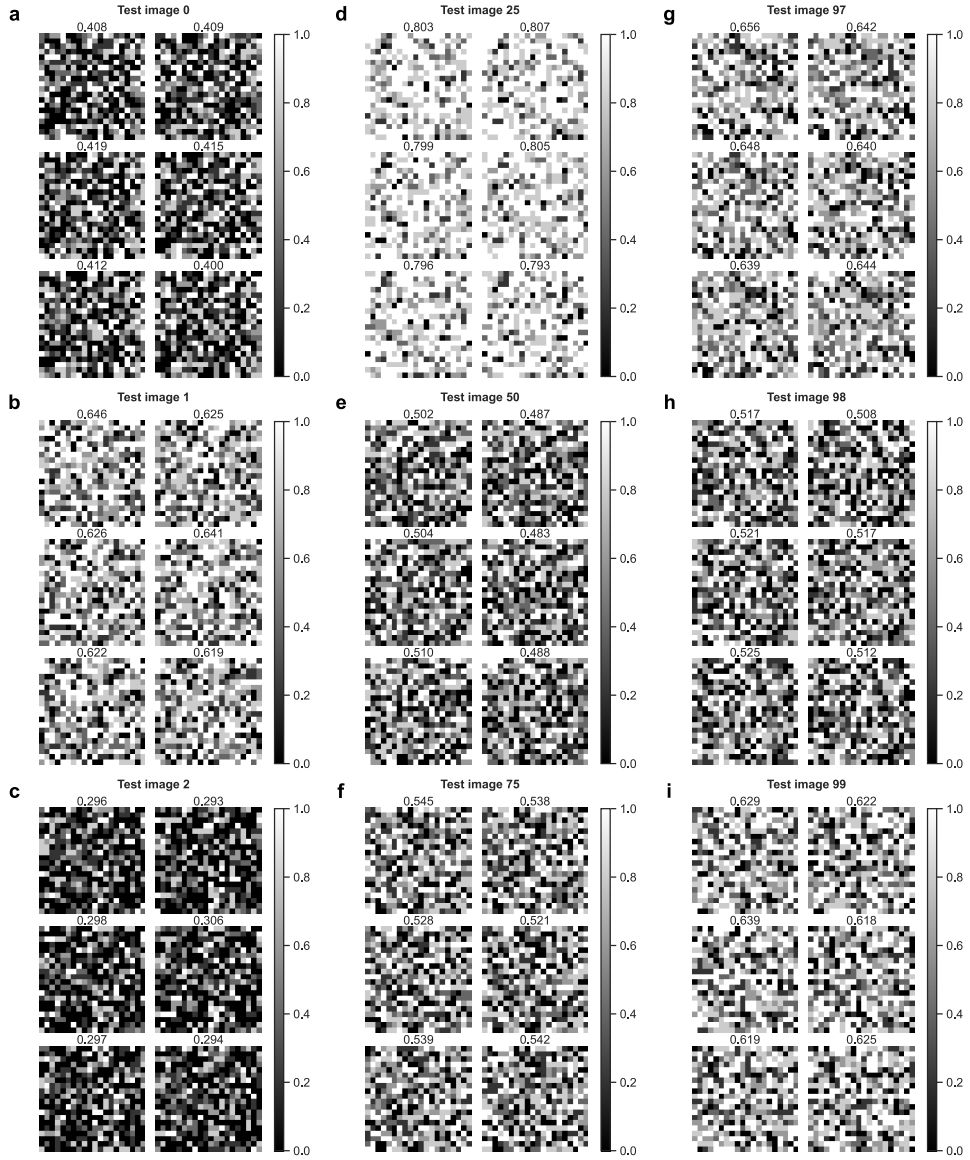


Figure 7.21: **Visualization of activation values on the SLM during the last layer experiment.** This figure displays the activations obtained from the data collected for the model of $N = 400$ hidden neurons and $K = 5$ shots per activation. The possible values for the 5-shot activations are 0, 0.2, 0.4, 0.6, 0.8, and 1. The activations of size 400 are rearranged into a 20×20 shape, which corresponds to their physical layout on the SLM. Panels **a** to **i** display the activations of test images with indices 0, 1, 2, 25, 50, 75, 97, 98, and 99, respectively, each with 6 repetitions of inference. The average value of the activations in each block is indicated at the top. The overall average activation value of the 100 test images is ~ 0.5219.

Exposure time	Photons per detection (pos.)	Photons per detection (neg.)	Total photons in output layer	Total detected photons in a full inference	Detected photons per MAC	Test accuracy
1.03 ms	79.9 ± 0.10	79.4 ± 0.10	1592.5 ± 1.5	2636.3 (0.98 fJ)	0.008 (0.003 aJ)	84.7 ± 3.2%
2.06 ms	159.9 ± 0.17	158.2 ± 0.20	3184.7 ± 2.5	4228.4 (1.57 fJ)	0.013 (0.005 aJ)	92.0 ± 2.3%
3.09 ms	239.7 ± 0.17	237.8 ± 0.22	4777.4 ± 2.6	5821.1 (2.17 fJ)	0.018 (0.007 aJ)	95.2 ± 2.0%
4.12 ms	320.2 ± 0.21	317.0 ± 0.20	6369.4 ± 2.5	7413.2 (2.76 fJ)	0.023 (0.009 aJ)	96.4 ± 1.8%
7.21 ms	560.1 ± 0.30	555.2 ± 0.33	11145.7 ± 5.1	12189.5 (4.54 fJ)	0.038 (0.014 aJ)	98.0 ± 1.3%
12.36 ms	960.1 ± 0.32	950.2 ± 0.40	19107.3 ± 4.1	20151.1 (7.51 fJ)	0.063 (0.024 aJ)	98.2 ± 1.1%
18.54 ms	1439.3 ± 0.56	1425.8 ± 0.41	28658.1 ± 6.6	29701.8 (11.08 fJ)	0.094 (0.035 aJ)	99.0 ± 1.0%

Table 7.6: **Optical energy consumption in SPDNN inference with varying photon budgets in the optical implementation of the output layer.** The first column displays the exposure time of the camera, which determines the number of detected photons. The average photons per detection for both positive (pos.) and negative (neg.) output are calculated from the 6000 dot products derived from 100 input images, 6 repetitions in the first layer inference, and 10 output nodes. The total photons in the output layer are determined by averaging 600 inferences of the last layer, each computing 10 output values. The total detected photons in a full inference are the sum of photons detected in both layers. The average photons per MAC is calculated by dividing the total number of MACs by the total detected photons. Standard deviations are calculated based on 30 repetitions of the last layer detection. The total detected number of photons in a full inference, along with the corresponding optical energy of photons at 532 nm, are displayed in the fifth column, with standard deviations omitted for simplicity. These results add the 1043.7 photons used in the first layer. The sixth column displays the average detected number of photons per MAC during a full inference, dividing the numbers in the fifth column by the total number of MACs, 317,600. The last column shows the test accuracy of the inferences at each photon budget.

the optical energy per detection. In order to perform the inference on the 100 input images and 10 output nodes, along with 6 repetitions of the activation values and 2 sets of weights, we need to perform a total of $100 \times 6 \times 10 \times 2 = 12000$ vector-vector dot products, each with a size of 400. Each vector-vector dot product detection is repeated 100 times. The results are presented in Table 7.6. The photons per detection of either positive or negative output are each averaged over $100 \times 6 \times 10 = 6000$ dot products. The total photons detected in the

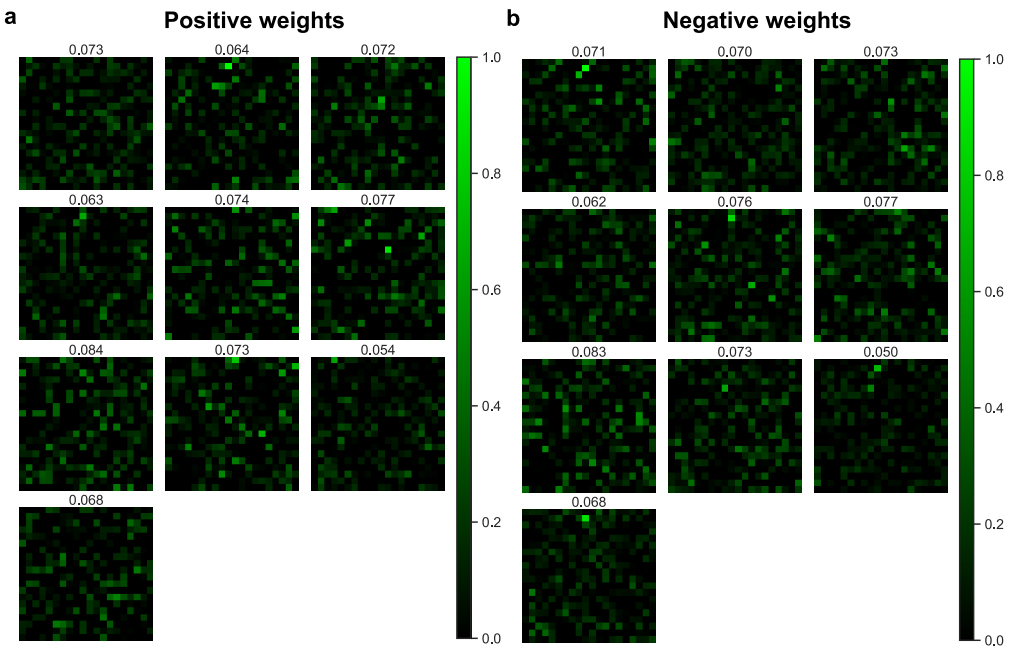


Figure 7.22: **Visualization of output layer weights on the OLED display.** This figure presents the positive and negative weights from a model with $N = 400$ hidden neurons and a weight matrix of shape 400×10. The positive and negative weights are depicted in panels **a** and **b**, respectively. To match the layout on the OLED display, each weight vector of size 400, corresponding to one of the 10 output nodes, was rearranged into a block with a shape of 20×20 and displayed using green pixels. The values were normalized to the range of 0 to 1 and the average value of each block is indicated at the top.

last layer per inference are averaged over the 100 input images and 6 repetitions, totaling $100\times6 = 600$ inferences. The standard deviation of the photon numbers are calculated based on the 100 repeated detections for each dot product. The total detected photons in a full inference is the sum of those in the last layer and the first layer. The average value of the binary activations collected for the $N = 400$ model is ~0.52186, resulting in a total of $0.52186 \times 400 \times 5 \approx 1043.7$ detected photons per inference in the first layer, with 5 shots per activation. This number is then combined with the total detected photons in the last layer to obtain the overall photon count for a full inference. We can see that the photon

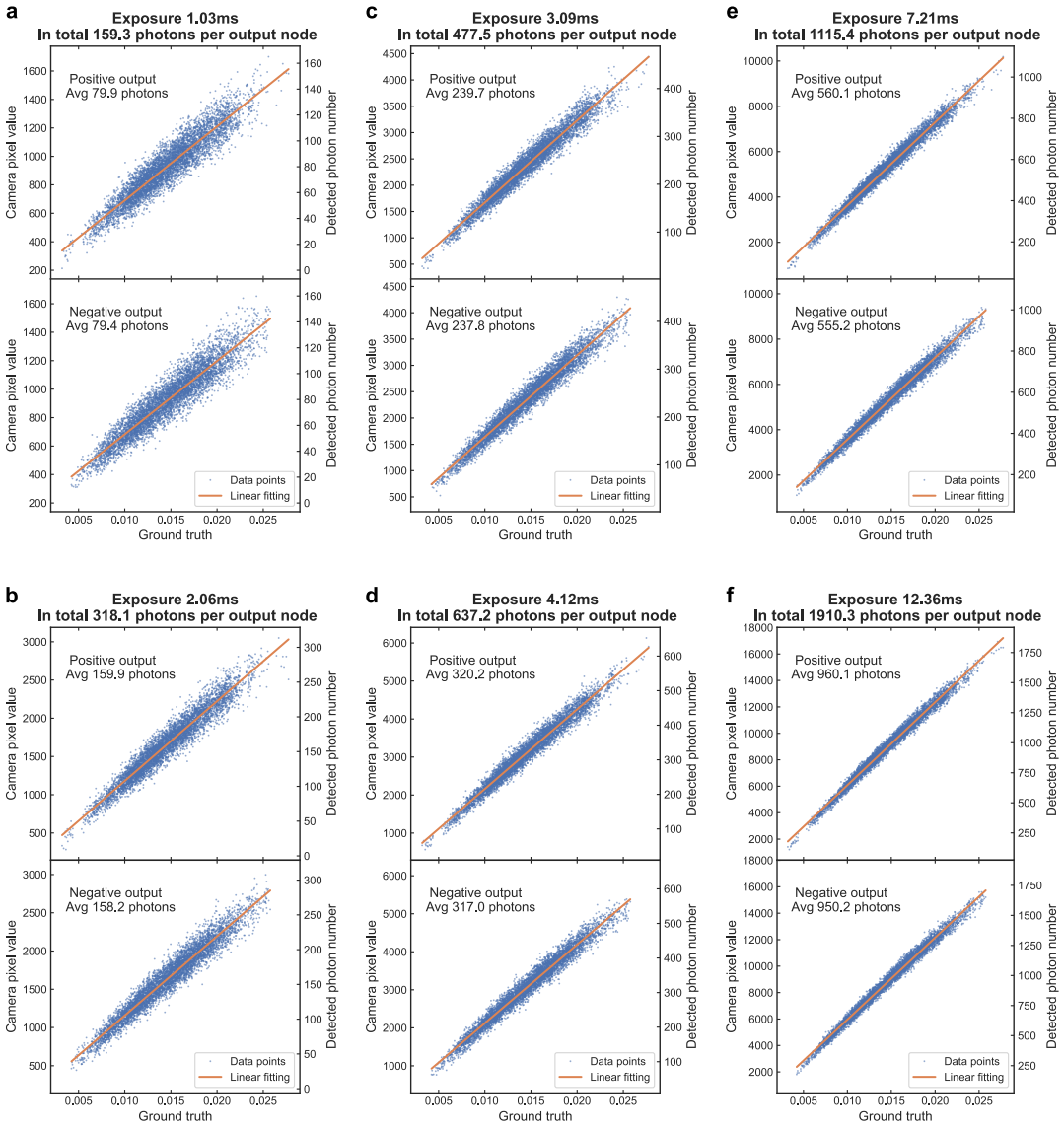


Figure 7.23: **Calibration of collected experimental data of the last layer inference.** This figure shows the raw data of "high-SNR" optical measurement on the qCMOS camera with various exposure times, each depicted in a separate panel. For each exposure time, one output value was obtained by measuring the output from both positive and negative weights. Each plot includes 6,000 data points, representing 100 test images, 6 repetitions in the hidden layer activation, and 10 output nodes. The ground truth values were computed using full-precision operations on a digital computer. Both the raw camera pixel values and the corresponding detected photon numbers are displayed on the y-axis, with the average detected photon numbers for the 6,000 data points noted in each plot.

budget can be reduced by 5 folds if we only have one shot per inference. In a full inference with $N = 400$ hidden neurons and $K = 5$ shots per activation, the total number of vector-vector products in the first layer is 400 and that in the last layer is 10 for the 10 output nodes. With dot products of size 784 in the first layer and 400 in the last layer, the total number of MACs in one inference process is equal to 317,600 ($400 \times 784 + 10 \times 400$). To calculate the number of detected photons per MAC, we divide the total number of detected photons in a full inference by the total number of MACs. The prediction of a given inference is made by directly evaluating the output values of each of the 10 output nodes. The output values are calculated as the difference between the positive and negative output intensity. The label of the node with the highest output value is then determined to be the predicted label. The test accuracy on the 100 test images is presented with its mean and standard deviation in the final column of Table 7.6. The standard deviation is determined by considering both the 6 repetitions of the first layer's inference and the 100 repetitions of detections in the last layer.

To visualize the impact of photon noise on accuracy in ONN inferences with a limited photon budget, the data collected from the last layer inference is depicted in Figure 7.23. In each panel, 6000 data points are plotted for either positive or negative output, considering the 100 input images, 6 repetitions in the first layer inference, and 10 output nodes. The ground truth dot product values are computed with high-precision operations on a computer. Both the raw camera pixel values and the corresponding photon count are shown on the vertical axes. As the number of detected photons per detection increases, the detected values become less noisy, resulting in a test accuracy that is closer to the ground truth of 99.2% (Table 7.5). Similar to conventional optical neural networks, the

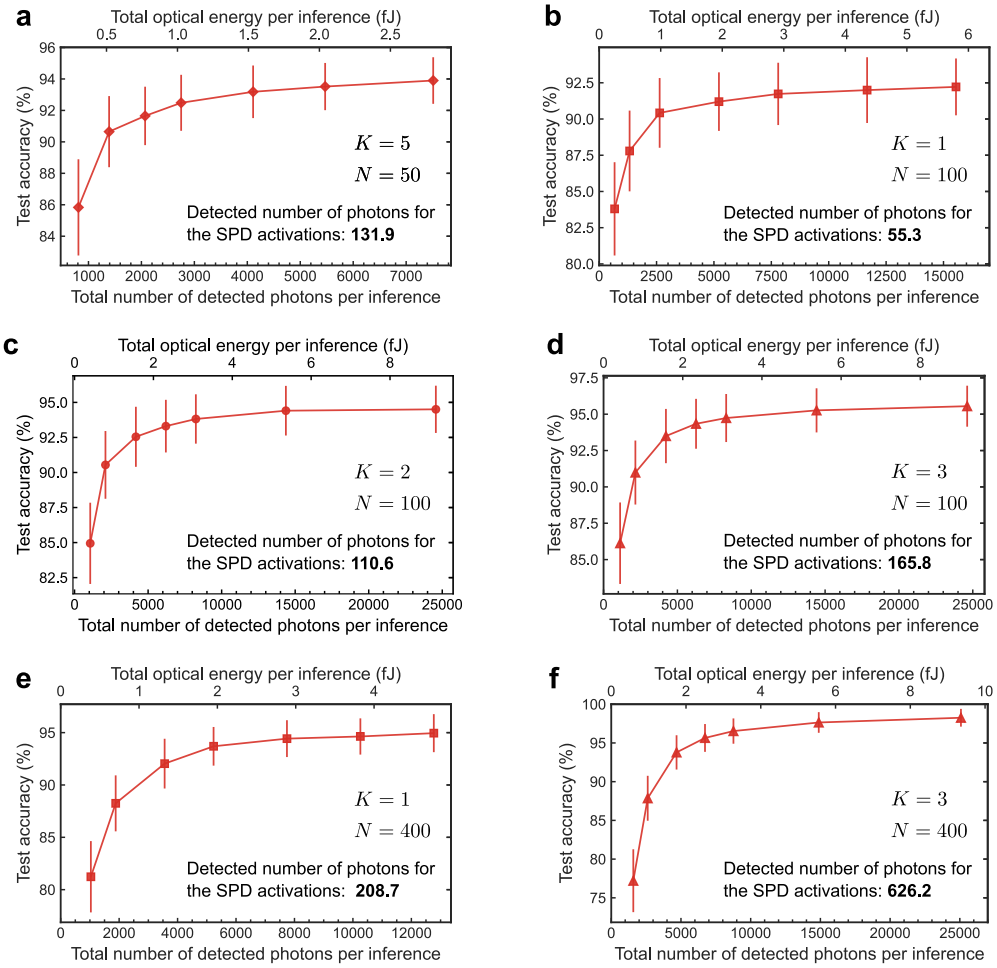


Figure 7.24: **Experimental results of full-optical implementation with different SPDNN configurations.** In addition to Figure 7.11b, this figure shows the results obtained with different numbers of hidden neurons (N) and shots of SPD measurements per activation (K). Each model uses experimentally collected activation values as input for the optical implementation of the output layer. The number of detected photons in the first $784 \rightarrow N$ layer to compute the SPD activations in each configuration is denoted in the corresponding plot. The noise-free test accuracies with full-precision output layer are shown in Table 7.5. The number of detected photons in the $N \rightarrow 10$ output layer is varied to control the noise in the optical implementation, which is reflected in the resultant test accuracy. The total number of detected photons per inference is the sum of the photon budgets in the two layers.

decrease in accuracy is primarily due to shot noise.

In addition, we performed the output layer optically for other configurations as well. The results are represented in Figure 7.24). The activation values collected in experiments of other choices of number of hidden neuron N and shots of SPD readouts K are used as the input for the output layer. If the output layer is implemented with full numerical precision, the test accuracies were shown in Table 7.5. These accuracies are the upper bound for the full-optical implementation with the presence of noise in optical implementation. For these configurations of numbers of hidden neurons (N) and shots of SPD measurements per activation (K), one inference through the $784 \rightarrow N$ hidden layer involves $N \times K$ SPD measurements to compute the activation vector in the hidden layer of size N. The detected number of photons for the SPD activation computation in the hidden layer of each configuration is denoted in the corresponding panel in Figure 7.24. The total number of detected photons per inference is the summation of this number and the total number of photons detected in the $N \rightarrow 10$ output layer, similar to the procedure we discussed above for the configuration of $N = 400$ and $K = 5$.

Similar to the plot in Figure 7.11b, the test accuracies increase with the detected optical energy in a similar trend to that of the $N = 400$, $K = 5$ we discussed in detail above. Comparing these panels with different configurations, we can see that models with a smaller number of neurons N exhibit greater resilience to noise when a similar number of photons are used in the output layer. For instance, comparing panels d and f, with approximately 2000 photons in the output layer (the second point from the left), the test accuracy declines more in the $N = 400$ model (panel f) than in the $N = 100$ model (panel d). Although the

models with smaller N and K suffer from a lower noise-free accuracy due to a smaller network size and higher stochasticity, as shown in Table 7.5. The final test accuracy is a combination of these two factors.

7.5 Additional Tests

7.5.1 Robustness to Experimental Errors

The first thing to check is the errors induced by the single-photon detectors. The two key parameters to consider when choosing commercial SPDs are photon detection efficiency and dark count rate. Photon detection efficiency refers to the amount of incident light that can be detected by the SPD. Although low photon detection efficiency is a common issue in many photon experiments, it does not add extra noise to our SPDNN models. This is because any attenuation to the light still follows a Poisson distribution and cannot be noisier than a single-photon detector. Hence, a low photon detection efficiency will only add to the overall transmission loss in the setup, and the input light power is usually redundant, so it will not affect the performance much. On the other hand, dark count rates, or false clicks, could pose a greater challenge in experiments with SPDs. False clicks are hard to distinguish from real signals, and the output of the detection is binary. The dark count rate of a functional SPD is typically between 10^{-5} and 10^{-2} false clicks per signal, depending on the experimental configuration. In some extreme circumstances, such as when the exposure time is very long or when it is hard to remove ambient light, the dark count rate could be as high as one false click in tens of detections, ruining the results of the experiment.

However, our SPDNN models are resilient to high dark count rates. As shown in Figure 7.25a, even with a false click in fewer than 10 measurements, we still obtain relatively good accuracy. The common range of $< 10^{-2}$ barely affects the performance of the SPDNNs.

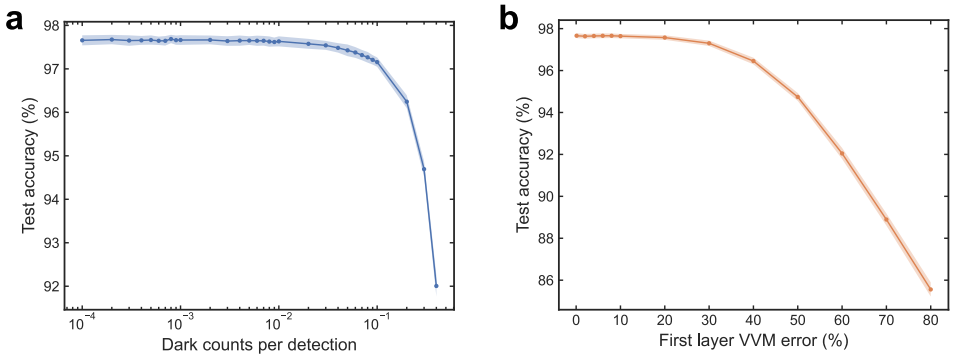


Figure 7.25: **Robustness tests for the SPDNN model. a,** MNIST test accuracy as a function of dark count rate of each SPD activation measurement. The dark count rate is varied to test the robustness of the SPDNN to noise in the measurement process. **b,** MNIST test accuracy as a function of the relative errors in the vector-vector multiplications (VVMs, or dot products) in the first layer. The errors are introduced at a fixed ratio with respect to the dot product result to simulate systematic errors in the optical setup. Both panels present results obtained from the $784 \rightarrow 400 \rightarrow 10$ incoherent model with one shot per inference (N = 400, K = 1), and the test accuracies are computed on the full test set of 10,000 images.

As introduced in Section 7.4.1, the dark count rate with our SPD setting is 0.01 per second per pixel. Given the exposure time of milliseconds, the effects due to dark counts are trivial in the experimental implementation. In summary, the robustness of SPDNN models to noise obviates the need for selecting specialized SPDs for experimental realization. Cost-effective SPDs can be employed for implementing SPDNNs with high performance. Furthermore, considering the significant power consumption of cooling systems for state-of-the-

art SPDs, relaxing the dark current requirement can greatly reduce the power consumption of the detection system.

The precision of linear operations is a crucial factor in neural network inferences. As discussed in Section 7.4.1, the accuracy of vector-vector multiplication may not be optimal when using a single-pixel camera for single-photon detection. To assess the effect of errors in dot product calculations on the performance, we conducted a simulation test by adding different levels of random noise to the dot product results in the first layer, which serve as the preactivations to the SPD activation function. The results, shown in Figure 7.25b, indicate that SPDNNs are robust to errors in linear operations, even with up to 20% relative noise. This robustness ensures the reliability of the experimental implementation.

7.5.2 Noise Resilience Compared to Conventional Models

In our SPD activation function, two key features set it apart from conventional neural networks: the quantization of activation values and the stochastic activation process. Both of these processes occur naturally through the detection of single photons. The intrinsic quantization of energy and detection process results in a nonlinear response to the input light intensity, eliminating the need for additional nonlinear operations in the neural network. This nonlinearity is evident in the higher MNIST classification test accuracy of SPDNNs compared to linear models. Additionally, the intrinsic photon noise in the activation function makes the output values stochastic. With more averaging, the stochasticity is reduced, resulting in a more precise output as seen in the implementation of

SPD activations in the fully-connected layers. This may imply that the noise is unwanted in the neural network inferences. However, the stochastic inference is inevitable in many real-world tasks with a physical device, our stochastic models demonstrated a high noise-resilience that can still yield reliable outputs regardless of this amount of stochasticity.

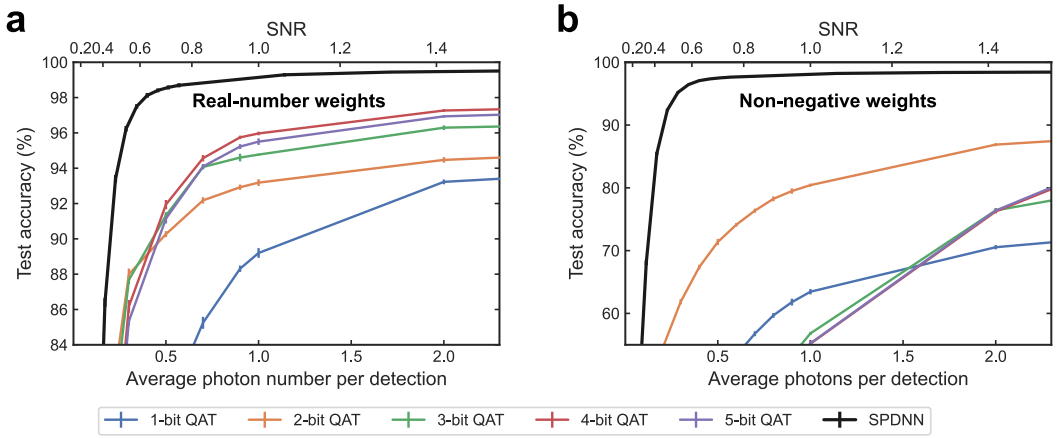


Figure 7.26: **Comparison of SPDNNs and QAT models on MNIST classification. a,** The test accuracy of MNIST classification using models with real-number weights is shown as a function of photon budget in the hidden layer. The SPDNN model employs the coherent SPD activation function, while the QAT models use ReLU as the activation function. **b,** The test accuracy of MNIST classification using models with non-negative weights is shown as a function of photon budget in the hidden layer. The SPDNN model employs an incoherent setup with non-negative weights, while the QAT models use Sigmoid activation function. The test accuracies are calculated on the full test set of 10,000 images. Both panels present results obtained from the $784 \rightarrow 400 \rightarrow 10$ model with one shot per inference (N = 400, K = 1), and the test accuracies are computed on the full test set of 10,000 images.

To evaluate the noise resilience of our SPDNNs against conventional continuous-variable models, we conducted experiments to compare the test accuracy of the models under varying levels of photon noise. We adopted quantization-aware training (QAT) as a popular noise-aware training method,

which involves quantizing the weights during training to make the model more noise-resilient. We trained deterministic QAT models with the same multi-layer perceptron (MLP) structure of $784 \to 400 \to 10$ and quantized the weight precision to a specific number of bits. We then compared the MNIST test accuracy of these models to SPDNNs with the same level of photon noise added during the neural network inference of the hidden layer.

For the real-valued QAT models that are compared to the coherent SPDNNs, we chose to use the ReLU activation functions. The QAT models adopted a deterministic quantization function and quantized the weights to the corresponding precision. During inferences, we performed computations with full precision, with the photon noise added to the pre-activation values of the hidden neurons. Figure 7.26a shows that the ReLU models exhibit high noise resilience, and harsh quantization does not significantly enhance the noise resilience but harms the overall precision. In fact, decreasing the quantization levels leads to decreased model performance at this photon noise level. The accuracy almost converges at a precision of 5 bits or higher.

For the non-negative QAT models that are compared to the incoherent SPDNNs, the non-negativity of the weights renders ReLU activation functions less effective. Hence, we use the Sigmoid activation function, more rigorously, the positive half of it, to train the QAT models. However, the models are not as noise resilient as with real-number operations, and stronger quantization is required to enhance the model robustness. As the simulation results show, the performance of models of precision 3 bits or more almost converges. It is worth noting that, despite having over 98% test accuracy without photon noise, the performance of these models with 3-bit precision or more is worse under such

noise levels. Decreasing the quantized precision is a tradeoff between noise resilience and overall accuracy. We observed that the 2-bit QAT model performs the best over other precisions. These results showed that all the QAT models are inferior to SPDNNs in terms of accuracy under the same or lower photon budget. This finding indicates that SPDNNs are more effective in achieving high accuracy in photon-starved environments.

Our results suggest that natural quantization of optical energy enhances noise resilience in neural networks, and that stochasticity could aid in searching for more accurate and noise-resilient models. However, we do not claim that the SPD activation function is the best way to train a noisy neural network, and we are open to exploring other noise-aware training methods that could further improve resilience. Our findings demonstrate that with appropriate training that takes into account the stochastic and quantized nature of optical energy in the realistic physical computing system, ONNs can achieve high performance even at very high noise levels, which was not previously possible. What makes it more intriguing about our approach is that it exploits the natural single-photon detection process.

7.5.3 Distribution of Expectation Values of SPD Activations

In this study, we explored the use of highly stochastic SPDNN models to achieve high performance in deterministic classification tasks. At first glance, this may seem counter-intuitive, as deterministic classification typically requires stable and reliable outputs, while stochastic models introduce inherent uncertainty. However, a closer examination of the characteristics of the activation values

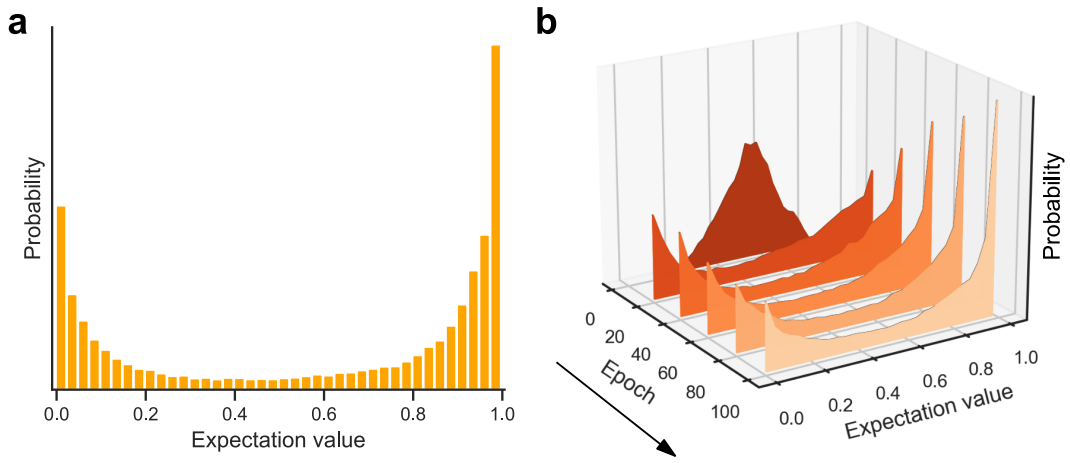


Figure 7.27: **Distribution of expectation values for hidden neuron activations.** **a,** The distribution of expectation values for each hidden neuron during the inference of a trained model. The model follows a multi-layer perceptron (MLP) structure of $784 \rightarrow 400 \rightarrow 400 \rightarrow 10$ using a coherent SPD activation function and real-valued weights (Appendix A.2). The expectation values of neuron activations in the first hidden layer are depicted. **b,** The evolution of the expectation value distribution during training. Different expectation value distributions of hidden neuron activations (as in **a**) are plotted at six training epochs to demonstrate changes throughout the training process.

in SPDNN inferences provides a more intuitive understanding of how this approach can achieve such high accuracy.

In Figure 7.27a, we present the distribution of expectation values for hidden neuron activations. This distribution is obtained using a single shot of SPD readout ($K = 1$). Since the activations are binary (either 0 or 1), the expectation value represents the probability of the activation being 1. We constructed this histogram by considering the inferences for all input images in the test set and all hidden neurons' activation values, so that the distribution is averaged over many different samples to show the overall picture of the general behavior of

the network inference. For example, a layer with 400 hidden neurons and 10,000 test input images would yield $400 \times 10,000 = 4 \times 10^6$ expectation values included in the histogram. We utilized an optimized SPDNN model with an MLP structure of $784 \rightarrow 400 \rightarrow 400 \rightarrow 10$ to generate this histogram, and we also found that this distribution is consistent across models with varying numbers of hidden neurons or layers, as well as coherent or incoherent SPD detection schemes.

Interestingly, we observed that the majority of neuron activations exhibit more deterministic expectation values rather than pure randomness. While some models trained with experimental limitations cannot reach absolute zero values, the peak at zero value shifts to a less sharp bump close to zero, still distributing towards either end rather than the middle value of 0.5. In Bernoulli sampling, an expectation value of 0.5 signifies that the probability of being 0 or 1 is equivalent, indicating that there is no useful information in the process, and the entropy is at its maximum. Noisy channels with such characteristics cannot carry valuable information for neural network inference. Consequently, during the training process, the model should strive to learn from the training set and update the neural network weights accordingly to capture the essential features. This process involves storing information in the trained model, which can be reflected by decreasing the entropy of each stochastic binary neuron.

In Figure 7.27b, we observe that as the model undergoes more training epochs, the expectation value distribution of activations becomes more concentrated towards 0 or 1. This indicates that the model retains more information and generates more reliable outputs.

However, it is important to note that while the entropy of each individual neuron decreases, at the network level, the average activation still tends to be

around 0.5 photons when considering all the neurons, denoting maximum entropy. This suggests that the neural network is effectively utilizing its capacity to extract information using all its neurons by increasing the overall network entropy. In fact, a network with all neurons having the same expectation value (entropy of 0) would not be able to learn any meaningful features.

In summary, while SPDNNs are inherently stochastic, the distribution of expectation values for hidden neuron activations leans towards deterministic outcomes, allowing the model to effectively learn features and achieve high accuracy in deterministic classification tasks. The training process shapes the probabilistic distribution of the neurons and allocates different neurons close to either 0 or 1 to learn the patterns of input images and output reliable inferences. Remarkably, the implementation of this allocation is exceptionally efficient in optical energy, as each activation only involves a photon click.

7.6 Conclusion and Discussion

Our research is an example of realizing a neural network using a stochastic physical system. Other ONN platforms can be readily adapted to use our approach by replacing the photodetectors typically used for readout of neurons at the end of a layer with single-photon detectors. The free-space matrix-vector multiplier (Chapter 3) used in our experiments is just one of many possible choices of architecture.

Beyond optics, our work is related and complementary to recent investigations in electronic, spintronic, and quantum neuromorphic computing [8, 251, 252, 253, 254, 255, 256, 257, 258], including training physical systems to

perform neural-network inference [41, 259, 260, 261, 262, 263, 264, 265]. Neural-network models are well-suited for analog devices due to their inherent tolerance for low-precision operations. While previous works [25, 266] have studied energy consumption in analog systems with lower but fixed precision, which is more akin to digital devices, our research moves beyond the fixed-precision assumption. Instead, we incorporate the stochastic processes inherent in realistic physical systems into neural-network modeling. Our results demonstrate that this *physics-aware probabilistic modeling* allows for significantly lower SNRs when optimized for specific tasks, thereby achieving much lower energy consumption compared to conventional noise-mitigation algorithms.

Noise, or uncertainty in signals, is a fundamental feature and the ultimate limit to energy efficiency in computing with all analog physical systems. It has long been realized that uncertainty is not always detrimental: not only does it not necessarily prevent accurate computation, but can in some cases even enable fundamentally new and more efficient algorithms or types of computation. Our work shows that using a quantum physical model of a particular hardware's inherent stochastic response at the software level can enable surprisingly large gains in energy efficiency.

The phenomena observed in our work seemingly relies on two key physical ingredients. First, the system's available states are effectively quantized, as in the photonic quantization of energy in our ONN demonstration, or the binarization that occurs in low-barrier, stochastic magnetic tunnel junctions [252]. Second, the noise in the system results in the quantized outputs of the system being stochastic. This suggests that ultra-low-SNR physical neural networks should be possible in many physical hardware platforms beyond photonics. Systems in

which shot noise dominates are natural matches with our approach and methods. Our approach could also be relevant to systems in which thermal (Johnson) noise dominates—as is typically the case in room-temperature electronics—but this will depend on not just the noise but also the system's dynamics. Which hardware platforms and system architectures can yield an overall energy benefit by being operated in a stochastic regime while maintaining computational accuracy is an important open question.

CHAPTER 8

IMAGE SENSING WITH ULTRA-LOW OPTICAL ENERGY

In the last chapter, we discussed SPDNN models enabled by physics-aware probabilistic modeling, which achieve high classification accuracy with very low optical energy—as low as ~1 photon per detection in the ONN systems. We also found that the detected photon energy required to determine the class of an object image can be as low as a few tens of photons per inference, far below typical expectations.

For example, in Section 7.4.3 (Table 7.5), we achieved a test accuracy of 92.8% ($N = 100$, $K = 1$) in digit classification by detecting an average of only ~ 60 photons in total using 100 neuron activation values measured by single-photon detectors. With only tens of photons detected, obtaining an image frame with sufficient SNR to discern the digit is impossible. This demonstrates that under very restricted photon budgets on the detector, such as limited integration time on the camera or fragile samples that cannot be exposed to bright illumination, operating the optical linear operation of an SPDNN model in the light field can significantly increase classification accuracy. In this chapter, we will further explore the possibilities of low-light sensing applications.

8.1 Background

8.1.1 Compressive Sensing and Single-Pixel Imaging

Compressive sensing is a framework that exploits the sparsity of signals to reconstruct them from a significantly reduced number of measurements [267].

This approach diverges from traditional methods by focusing on the underlying structure of the signal rather than the raw data volume, thus enabling efficient and effective signal reconstruction.

Single-pixel imaging [268], also known as computational ghost imaging, is an innovative application of compressive sensing that reconstructs images using a single photodetector rather than a high-resolution sensor array. This technique is particularly advantageous in scenarios where traditional imaging sensors are impractical or too costly.

Single-pixel imaging involves illuminating the scene with a sequence of known patterns, such as random binary patterns or Hadamard patterns. Each pattern is projected onto the scene, and the light reflected from the scene is measured by a single photodetector. For each illumination pattern, the single-pixel detector captures a single intensity value, which represents the aggregated light intensities modulated by the pattern. This process is repeated for a series of patterns to collect sufficient data for image reconstruction.

The measurements obtained from the single-pixel detector are utilized to reconstruct the image using compressive sensing algorithms. Given that the patterns and measurements are known, the reconstruction algorithm can solve for the original image by identifying the sparsest solution that matches the measured data.

Compressive sensing and single-pixel imaging have transformed the fields of signal processing and imaging by enabling high-quality reconstruction from minimal measurements. By leveraging the sparsity of natural signals and utilizing advanced computational algorithms, these techniques offer efficient and

effective solutions for data acquisition and processing.

8.1.2 Image Classification with Pattern Illumination

In many applications, such as anomaly detection and cell cytometry, the primary goal is to determine the class or category of an object rather than acquiring detailed spatial information. In these cases, achieving high-resolution spatial intensity distribution is unnecessary. Instead, a more efficient approach involves using pattern illumination combined with advanced computational techniques to classify objects with minimal measurements.

Pattern illumination involves projecting a sequence of designed patterns onto the object and capturing the resultant light interactions. By carefully selecting and optimizing these patterns, sufficient information can be extracted for classification without the need for full image reconstruction. This method leverages principles similar to those in compressive sensing and single-pixel imaging but focuses on classification.

The primary benefit of using pattern illumination for classification is the significant improvement in sensing efficiency. By reducing the number of measurements and focusing on classification, the system conserves resources and operates faster. This method is particularly advantageous in low-light conditions or when the power budget is constrained, as it minimizes the need for extensive illumination and data processing. Moreover, this approach could also enhance system robustness, as classification relies on key features extracted through pattern interaction rather than detailed spatial data, making it less susceptible to noise and distortions.

Recent advancements in computational processing in the optical field have further optimized the information flow before the detection bottleneck [269, 270, 271]. Among these, end-to-end optimization through neural-network computation shows promise with the rapid advancement. This concept can be referred to as a *"neural sensor"* [269], which integrates neural network processing with sensor technology to enhance classification accuracy and efficiency.

8.1.3 Image Sensing with Limited Optical Energy

Low-light applications are crucial in various fields, including medical imaging, astronomy, surveillance, and environmental monitoring. The challenge of working with limited optical energy necessitates innovative approaches to ensure efficient and accurate image sensing.

The need to operate under strict power budgets in portable and battery-operated devices, such as medical imaging equipment and remote sensors, necessitates efficient low-light sensing techniques. High-intensity light can damage samples or reduce the effectiveness of fluorescent markers in biological imaging. Low-light environments, such as deep space or underwater, inherently restrict the available optical energy. By focusing on classifying objects with minimal measurements, it is possible to achieve high detection efficiency. Pattern illumination combined with advanced computational techniques can extract sufficient information for classification without the need for detailed image reconstruction, thus enhancing sensing efficiency in low-light conditions.

8.1.4 Direct Imaging vs. Image Sensing under Restricted Optical Energy

Direct imaging and image sensing under restricted optical energy differ significantly in their approaches and effectiveness, particularly in low-light conditions. Direct imaging involves capturing a high-resolution image of the object and then applying digital processing to the captured image. This approach faces challenges when optical energy is limited, as the detection signal-to-noise ratio (SNR) decreases, adversely affecting imaging accuracy.

In low-light scenarios, direct imaging requires longer exposure times or higher sensitivity sensors to capture sufficient light, introducing noise and artifacts that reduce image quality and accuracy. High-resolution images demand substantial computational resources for processing and analysis, which can be inefficient under power constraints. By applying operations in the optical field first and using a few optimized optical measurements, it is possible to achieve better classification accuracy with the same optical energy. Pattern illumination techniques enhance the SNR by focusing on key features necessary for classification rather than capturing detailed spatial information. Advanced computational techniques, including machine learning, can optimize the classification process, making it more robust to noise and distortions.

Comparing the two approaches highlights the advantages of pattern illumination for image classification under restricted optical energy. While direct imaging struggles with low SNR and computational inefficiencies in low-light conditions, pattern illumination and advanced computational techniques offer a streamlined solution for efficient and accurate object detection and classifi-

cation. This method significantly improves sensing efficiency by reducing the number of measurements needed and focusing on the critical task of classification, making it ideal for applications where optical energy is limited.

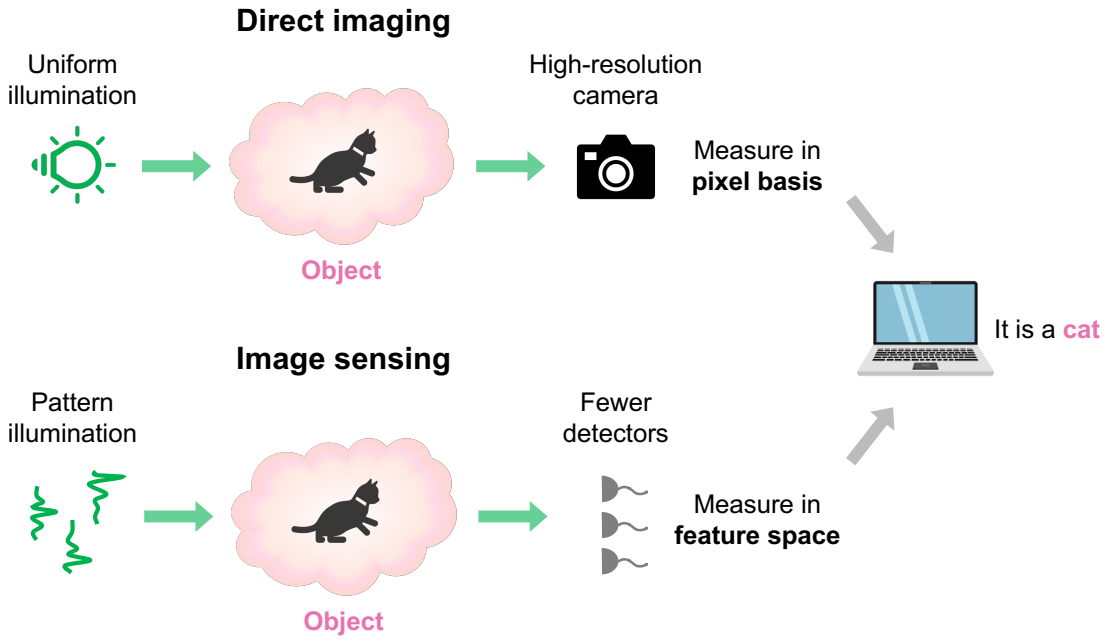


Figure 8.1: **Diagrams of direct imaging and image sensing.** The top diagram illustrates the direct imaging scheme, where the object is illuminated uniformly and captured by a high-resolution camera, measuring in the pixel basis. This image is then processed by a computer to classify the object, such as determining that the object "is a cat." The bottom diagram shows the image sensing scheme, where the object is illuminated with specifically designed patterns. The detected features corresponding to the illumination patterns, rather than the pixel values in the image, are then processed by a computer to classify the object. This method requires fewer detectors, which measure the scene in feature space rather than pixel basis. The green arrows denote the optical signals, while the grey arrows denote the electrical signals.

8.2 Task-Specific Incoherent Image Sensing Using Single-Photon Detectors

Building on the SPDNN models introduced in Chapter 7, it is natural to extend their application to task-specific incoherent image sensing. As detailed in Section 7.3.1, we assume the optical system employs incoherent light to encode and modulate information through light intensity.

The linear operation achieved by optical linear processors is fundamentally realized through the natural propagation of light, which is analogous to light behavior in any image sensing setup. In this section, we focus on an incoherent setup where linear operations are performed via the attenuation of light intensity.

8.2.1 Incoherent Single-Photon Detection (SPD) Neural Sensors

Consider the scenario where input images are static objects that passively attenuate light. The pixel values of these images range from 0 to 1, determining the transmission rate of the corresponding spatial mode. A pixel value of 1 indicates that the object is completely transparent to the light signal, while a pixel value of 0 signifies that the object is entirely opaque and blocks all light. For pixel values between 0 and 1, the object partially attenuates the light passing through it. For example, an image with a resolution of $d = 28 \times 28 = 784$, the input image can be represented by an input vector $\vec{x}$ of size 784. This setup is typical of any

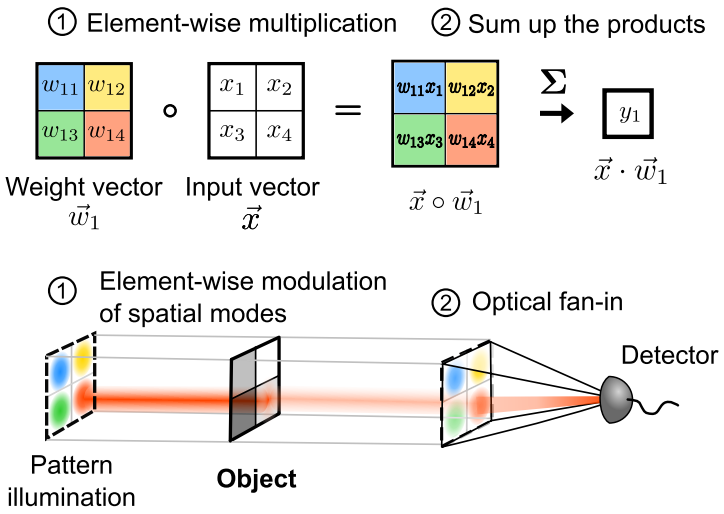


Figure 8.2: **Diagram of dot product operations in the incoherent single-photon detection (SPD) neural sensors.** In this setup, the object in image sensing passively attenuates incoming light. Weight vectors are encoded as illumination patterns that shine on the object. The attenuated light then undergoes an optical fan-in process, completing the dot product operation.

object located on the image plane of an incoherent imaging system.

As illustrated in Figure 8.2, we apply a pattern illumination represented by another vector $\vec{w_1}$ of the same size of $\vec{x}$. Each element in the vector $\vec{w_1}$ represents the light intensity of the corresponding spatial mode. When the pattern vector$\vec{w_1}$ propagates through the object represented by $\vec{x}$, it undergoes element-wise multiplication by the attenuation of the object, resulting in $\vec{w_1} \circ \vec{x}$. The optical fan-in process then sums up the optical intensity detected by the detector, effectively computing the dot product result $y_1 = \vec{w_1} \cdot \vec{x}$.

This process is similar to what's shown in Figure 3.1, only with $\vec{x}$ and $\vec{w_1}$ exchanged. In this setup, the input vector no longer requires a fan-out process as it remains static and passively attenuate the incoming light pattern represented by the vector $\vec{w_1}$. In the context of neural networks, $\vec{w_1}$ represents one row of the

weight matrix W in the linear operation.

For a specific classification task, the incoherent SPDNN model is trained as described in Section 7.2 and Appendix A.1. Assume we use a simple MLP architecture as detailed in Section 7.3.1, with the weight matrix of the first linear layer W having dimensions $N \times d$, where N is the number of neurons in the hidden layer and d is the dimension of the input vectors. To perform the first linear operation of the SPDNN model, $\vec{y} = W\vec{x}$, we need to implement N different weight vectors $\vec{w_i}$ for $i = 1, 2, \cdots, N$.

8.2.2 Pattern Multiplexing Schemes

In the image sensing scheme, since the input vectors now represent passive attenuations, the absence of the fan-out process for the input vectors necessitates some form of multiplexing of the weight vectors $\vec{w_i}$ to complete the full linear operation. This can be achieved practically by implementing the weight vectors sequentially over time or encoding them at different frequencies and detecting them simultaneously with different detectors, as illustrated in Figure 8.3. This multiplexing approach allows the system to handle the linear operations efficiently, maintaining the benefits of incoherent single-photon detection while facilitating complex image sensing tasks.

Time Multiplexing Scheme The time multiplexing scheme is a prevalent choice in compressive sensing and single-pixel imaging [**?**], as well as in some ONN implementations [24, 16]. In this scheme, the illumination patterns are sent sequentially. This method efficiently implements optical linear operations

with a relatively simple experimental setup. Although there might be concerns about speed loss in certain situations, the high sampling rate of SPDs generally makes this approach feasible for most applications.

Frequency Multiplexing Scheme The frequency multiplexing scheme is also common among ONN implementations [16, 21, 60]. A nuance here is that unlike optical computing techniques such as wavelength-division multiplexing (WDM) [21], sending illumination patterns of different frequencies to the same target may suffer from dispersion. However, as demonstrated in Section 6.2, our single-photon spectrometer with approximately 400 modes spans a wavelength range of only roughly 590 nm to 650 nm (Figure 6.4). This result indicates that our SPD neural sensors, which typically only have tens of neurons, should not be significantly affected by dispersion. Besides, ONNs are generally believed to be robust to dispersion [46], so we do not anticipate significant issues in practice. Furthermore, for situations where sensing speed is a critical concern, frequency multiplexing is particularly advantageous due to its capability for simultaneous measurement.

8.2.3 Estimation of Optical Energy Budgets

In this section, we detail the methodology for estimating the optical energy budget in the implementation of the SPD neural sensors. During the inference phase, we set the slope variable η to 1, such that the value of the optical energy λ is measured in units of photon number (see Appendix A).

The total *detected* photon energy is estimated directly from the activation val-

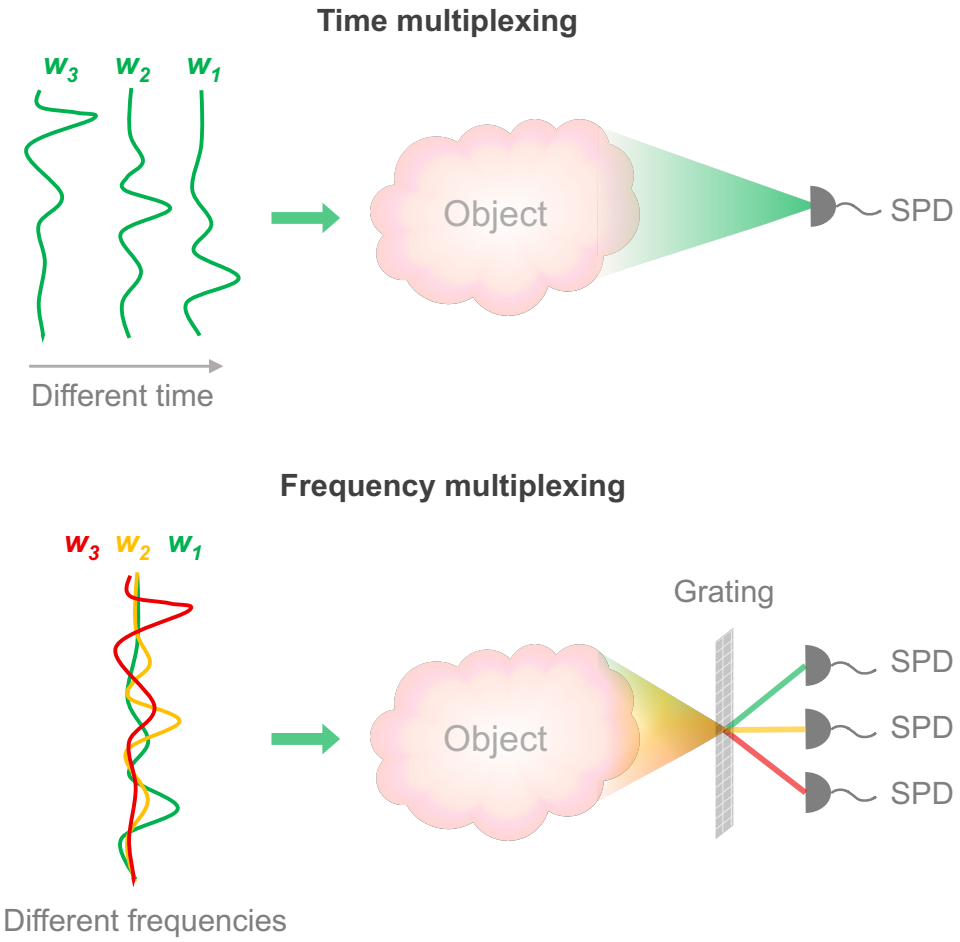


Figure 8.3: **Multiplexing schemes for the single-photon detection (SPD) neural sensors.** In the time multiplexing scheme (upper row), the illumination patterns (weight vectors) are sent to the object at different times but at the same frequency. The readouts on the single-photon detector (SPD) at the corresponding times are recorded as the activations of the respective SPD neurons. In the frequency multiplexing scheme (lower row), different illumination patterns (weight vectors) are sent simultaneously but at different frequencies. Before detection, a grating separates the different frequencies, which are then processed to correspond to the activations of the respective neurons.

ues in the hidden layer. For a given input image $\vec{x}$ and the first layer weight matrix W, the activation vector in the hidden layer is determined by $\vec{a} = f_{\text{SPD}}(W\vec{x})$. Since f_{SPD} is a probabilistic sampling function, the actual values of $\vec{a}$ of each repetition of inference are random variables. Besides, for K shots of SPD measurement per activation, each activation value of neuron index i is an average of K repetitions of the binary SPD readout values, $a_i = \frac{1}{K}\sum_{k=1}^{K} a_i^{(k)} \in \{0, 1/K, 2/K, \ldots, 1\}$ (see Section 7.2.2, Algorithm 3 in Appendix A.1). Therefore, the total number of detected photons for a given input image $\vec{x}$ is $\mathbb{E}_{\vec{a}}\left[K\sum_{i=1}^{N} a_i\right]$. For the entire test dataset, the total detected number of photons per inference must be averaged over all test images in the dataset, resulting in $\mathbb{E}_{\vec{x}}\left[\mathbb{E}_{\vec{a}}\left[K\sum_{i=1}^{N} a_i\right]\right]$, where $a_i = f_{\text{SPD}}(\vec{w_i}\cdot\vec{x})$ and $\vec{w_i}$ is a row in W.

In some applications, such as photobleaching in biomedical imaging, the concern is more about the *illumination* optical energy delivered to the samples rather than the detected energy. In the SPD neural sensors, the pattern illumination is determined by the weight matrix W. The input images have values in the range $[0, 1]$, representing intensity attenuation. During the training process, the values of the weight matrix W are optimized and restricted to non-negative numbers due to the incoherent setup, but no maximum value constraint is applied. Therefore, the value range of W is $[0, w_{\max}]$. Since the slope variable η is set to 1 during inference (see Algorithm 3 in Appendix A.1), the value of an element in W represents the illumination optical energy in units of photon number. Thus, the total number of photons used for illumination per inference is the sum of all elements in W. Considering the K shots per activation, the total number of illumination per inference is estimated as $K\sum_{i=1}^{N}\sum_{j=1}^{d} W_{ij}$.

This total illumination is a constant value for any input image, determined

solely by the illumination patterns represented by the optimized weight matrix W. The entire image sensing setup, including the illumination light field and detection scheme, remains consistent throughout the experiment. The single-photon detectors (SPDs) continuously collect data over time, and the output values change dynamically as the object changes.

8.2.4 Direct Imaging with Set Detected Optical Energy

In the previous sections, we introduced the SPD neural sensing scheme and the methodology for estimating photon budgets. To demonstrate the advantages of SPD neural sensing in terms of photon efficiency, we must compare its performance with that of the direct imaging scheme under the same optical energy constraints.

As shown in Figure 8.1, the direct imaging scheme represents the conventional imaging process, wherein an image of the object is captured. Uniform illumination light is used to shine on the object, and an image is recorded on the image plane. However, with limited optical energy upon detection, the resulting image can be very noisy due to photon noise (Figure 7.1b) and other noise sources such as camera readout noise. We set a specified value for the total number of detected photons per image, equivalent to the sum of photons detected across all pixels in the image. As illustrated in the inset panels of Figure 8.4, the SNR in the object images scales with the set value of detected optical energy per image. As expected, the higher the detected optical energy, the less noisy the image becomes. These noisy images are then used as inputs to a neural network on a digital computer to perform the image classification task.

Because the illumination is uniform, the illumination optical energy per image in the direct imaging scheme can be estimated by dividing the detected optical energy per image by the average attenuation of the object, which corresponds to the average value of the input vector. To obtain the average illumination optical energy per inference across the entire test dataset, we simply divide the detected optical energy by the average value of every input image in the test dataset.

In contrast, SPD neural sensing employs illumination patterns that are specifically optimized for the given classification task. This optimization serves two primary purposes:

- Reducing the number of detections required, and
- Optimizing the light field to mitigate the information loss due to the uncertainty inherent in the highly stochastic photon detection process.

Therefore, we expect that SPD neural sensing will more effectively extract useful information from the light field for image classification, despite the highly stochastic nature of photon detection when the detected optical energy is very low.

8.2.5 Performance Comparison

By comparing these two schemes, we aim to highlight the enhanced photon efficiency and robustness of SPD neural sensing in low-light conditions, demonstrating its potential for applications requiring high classification accuracy with minimal optical energy.

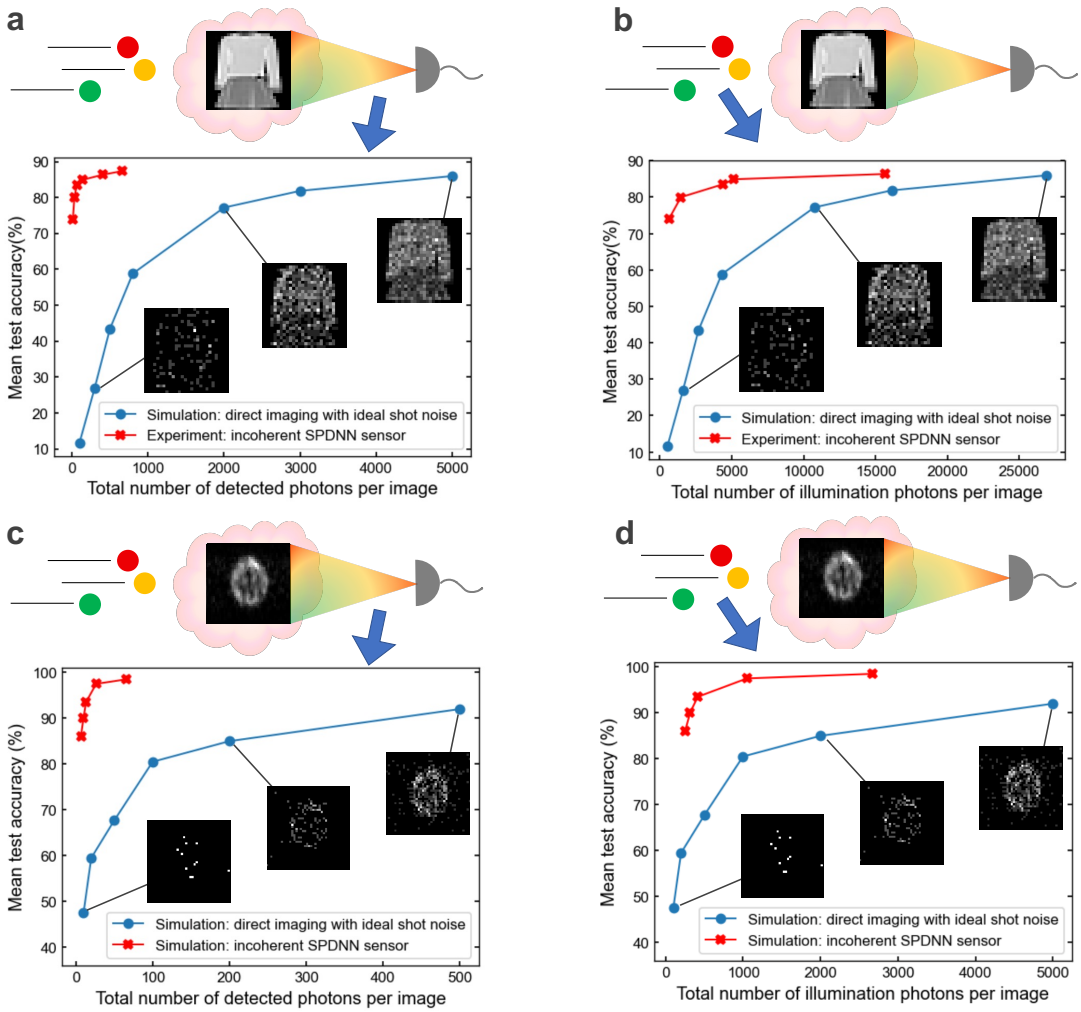


Figure 8.4: **Test accuracy with different detected and illumination photon budgets.** The figure presents a comparison of mean test accuracy as a function of the number of detected (left column) and illumination (right column) photons per image for both direct imaging and incoherent SPDNN sensing for Fashion-MNIST (upper row) and cell classification (lower row) tasks. In all subfigures, the blue curves represent simulated results for direct imaging with ideal shot noise, and the red curves show the experimental results for the incoherent SPDNN sensors with different number of hidden neurons N.

We present the results of two classification tasks: the benchmark FashionMNIST dataset and a flow cytometry dataset for cell classification. As detailed in Sections 8.2.1 to 8.2.4, we trained the incoherent SPDNN models using a multilayer perceptron (MLP) architecture with one hidden layer, as described in Section 7.3.1. For the FashionMNIST classification task, we collected experimental data from the trained SPDNN models using the setup introduced in Section 7.4. The results are shown in Figure 8.4.

In Figure 8.4, the left column (panels **a** and **c**) shows the mean test classification accuracy with varying detected optical energy, while the right column (panels **b** and **d**) shows the test accuracy with varying illumination optical energy (Section 8.2.3). The upper row presents the results for FashionMNIST, where the SPDNN models have varying numbers of hidden neurons (N = 10, 25, 50, 100, 300, 500). The lower row presents the results for cell classification, with models having different numbers of hidden neurons (N = 10, 15, 20, 40, 100). Throughout all experiments, the number of shots per activation (K) was set to 1 (1 shot per activation), and variations in the photon budgets of SPDNN models were solely due to differences in the number of hidden neurons N.

The direct imaging results were estimated entirely through simulation by adding ideal shot noise to the images corresponding to the respective photon budgets. This simulation serves as an upper bound for practical scenarios, as images collected in realistic experiments would typically exhibit more noise than this ideal case. This approach slightly biases the results in favor of direct imaging, allowing us to clearly illustrate the absolute performance gap between the two schemes.

The experimental results demonstrate that SPD neural sensing consistently

outperforms direct imaging under identical optical energy constraints. This superior performance is attributed to the optimized illumination patterns in SPDNN, which efficiently utilize limited photons for accurate classification. In low-light conditions, where photon budgets are severely restricted, SPD neural sensing maintains high classification accuracy, proving its robustness and efficiency.

These findings underscore the potential of SPD neural sensing for various applications that require precise image classification with minimal optical energy. By leveraging the advantages of optimized illumination and efficient photon usage, it provides a compelling alternative to traditional direct imaging methods, particularly in scenarios demanding high sensitivity and low-light operation.

8.3 Real-Time Sensing of Moving Objects

In addition to the performance advantage over direct imaging, we also explore the practical application of SPD neural sensors in real-world scenarios. Many low-light sensing applications, such as night vision and biomedical imaging, involve objects that are not static but moving at high speeds. The ability to capture fast-moving objects is therefore essential for the realistic application of SPD neural sensors. Here, we demonstrate the real-time sensing capabilities of our SPD neural sensors.

8.3.1 Constant Illumination Field

As discussed at the end of Section 8.2.3, our SPD neural sensors maintain a constant sensing field while operating. The sensing scheme does not need to change while the detectors consistently collect data dynamically. The spatial calibration of the illumination patterns determines the field of view (FoV) of the sensor. Since the illumination is confined to the FoV, objects outside this area are not perceived by the SPD neural sensor.

Using the frequency multiplexing scheme as an example (the time multiplexing scheme should exhibit similar performance, though with a reduced maximum speed due to the time multiplexing factor), each illumination pattern or weight vector is encoded in the light field at its corresponding frequency and detected by the respective detector. Once calibrated, the intensity level of the light field, the imaging system, and the detector integration time remain constant throughout the sensing period.

8.3.2 Continuous Data Collection

Although the experimental settings remain unchanged, the detectors continuously collect data with a given integration time. The data collection speed is limited only by the sampling rate of the SPDs, which can reach up to 100 GHz [272]. This rate also defines the time resolution for the real-time image sensor. Each detector, representing a neuron in the hidden layer, continuously collects data (a click or no click) with a fixed integration time. For each time cycle, the N readouts from the detectors form a neural activation vector of length N ($K = 1$ for all image sensing applications). These detected signals are then transferred

to a digital computer to perform a small linear operation, resulting in an output vector. With the number of hidden neurons N typically in the tens and the number of classes usually a few, the size of the linear operation remains manageable, not exceeding a few hundred parameters. This linear operation produces the output vector for each time cycle, determining the classifier's prediction in real time.

8.3.3 Demonstration of Real-Time Sensing

To demonstrate the real-time sensing of moving objects, we use the flow cytometry dataset for cell classification (Section 8.2.5). A typical transit time for cell sorting can be in microseconds [273], which is well within the range of the SPD neural sensor. Besides the five classes, we added a "Null" class to indicate no cell detected. Training samples of empty and largely cropped frames were included to fit this "Null" class.

For the demonstration, we used an incoherent SPDNN model with $N = 50$ hidden neurons and trained it for this six-class classification task, including the "Null" class. In the simulation, we spatially shifted the input images to emulate the process of cells moving into the FoV of the SPD neural sensor. The ground truth input images at different phases of entering the FoV, along with the corresponding real-time output vectors, are shown in Figure **??**. The FoV images from left to right demonstrate the process of the cell gradually entering in the FoV of the incoherent SPD neural sensor. Before the cell enters the FoV, the output amplitude of the "Null" class is the highest. As the cell fully enters the FoV, the neural sensor correctly detects the cell class from the third image onward.

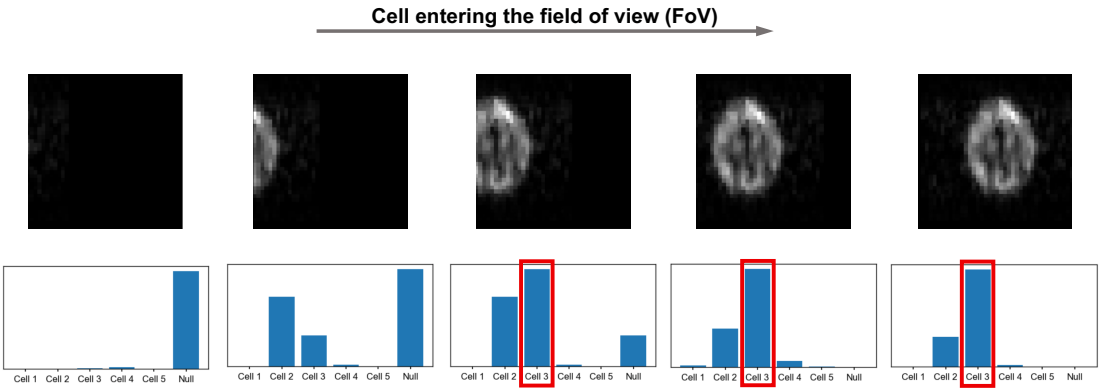


Figure 8.5: **Demonstration of the object cell entering the field of view (FoV) of the single-photon detection (SPD) neural sensor.** From left to right, a cell gradually enters the FoV of the incoherent SPD neural sensor with $N = 50$ hidden neurons. The corresponding output vectors are displayed below the FoV images. The vectors were normalized by the Logmax function for better visualization. The red boxes indicate correct detection of the cell by the image sensor.

This demonstration highlights the capability of SPD neural sensors to perform real-time sensing and classification of moving objects, showcasing their potential for practical applications in dynamic and low-light environments.

8.4 Discussion

In this chapter, we demonstrated that the incoherent SPDNN models can significantly enhance image sensing accuracy even with highly restricted photon budgets. Compared to directly capturing an image of the object, our SPD neural sensor achieves higher accuracy with the same amount of optical energy detected or illuminated. This approach has potential for various low-light sensing applications, such as biomedical imaging, where illumination light is limited due to concerns like photon bleaching [274].

The high sampling rate of single-photon detectors enables rapid inference in SPD neural sensors, making them highly suitable for detecting fast-moving objects, especially under very low photon budgets. Even in scenarios where the optical power is not extremely low, high sampling rates and short integration times, as required for fast-moving objects, etc., can result in low detected optical energy as well. In these situations, our SPD neural sensors demonstrate reliable performance due to their high efficiency with the available photon energy.

As discussed in Sections 7.4.2, 7.5.1 and 7.5.2, our SPDNN models exhibit excellent robustness to various sources of fluctuations, noise, and errors. This robustness translates to the corresponding incoherent SPD neural sensors, ensuring reliable implementation and operation. The inherent resilience of SPDNN models to such perturbations contributes significantly to their practical applicability.

Furthermore, as shown in Section 7.3.2, coherent SPDNN models with real-number weights exhibit superior performance compared to their incoherent counterparts. Therefore, implementing SPD neural sensors using coherent setups is expected to yield even better performance. Additionally, ONN platforms with coherent setups [12, 57, 13, 56, 63, 17, 16] typically have the capability to implement arbitrary unitary operations (i.e., complex-number weights). This capability can be further exploited for efficient image sensing in complex coherent light fields, such as imaging through diffusing media.

In conclusion, the SPD neural sensors, leveraging both incoherent and coherent SPDNN models, offer a promising solution for low-light sensing applications, especially in low-light and dynamic environments. Their ability to maintain high accuracy and robustness under low photon budgets and rapid in-

ference requirements makes them a versatile tool for advanced optical sensing technologies.

CHAPTER 9

FUTURE DIRECTIONS

9.1 On-Chip Implementation

One significant shortcoming of the projects presented in this thesis is that they were all implemented with bulky free-space optical components. An obvious solution is to integrate these components onto a chip.

Integrating the approach presented in Chapter 7 onto a chip is straightforward, as it does not require a specific specific physical realization of the optical linear operation. The existing on-chip ONN platforms [51, 12] are readily compatible to the applications, with the only major concern being a sufficient system size.

For the work reported in Chapter 6, there are also numerous opportunities for improvements using integrated, on-chip devices [275, 276]. For example, one could implement the combined DOPA and AFC source on a single chip, with single-spatial-mode waveguiding; this could eliminate detrimental spatio-temporal effects in AFC. It would allow for longer interaction lengths than in free space (limited by Rayleigh length) and allow for high intensities with less power, both resulting in more powerful unitary control. Combining the DOPA and AFC processes into one device would also eliminate most interface losses of the quantum light prior to detection. Using integrated photonics would also allow the engineering of dispersion to optimize the squeezing bandwidth, the number of modes in the DOPA, and the interaction complexity imparted by AFC pump. Frequency-encoded squeezed light is not restricted to using su-

permodes of a continuous basis: quantum states can be engineered to inhabit a (more conventional) basis of (ultrafast) frequency combs lines or discrete frequency bins [158, 277, 278, 279, 280], through dispersion and pump engineering.

9.2 Beyond the Current Capability

Another natural future work could explore the extension of our research to neural networks with larger sizes (wider and more layers, which could both improve the capability of the neural network and further amortize the energy cost of the final, high-SNR layer, if used), more sophisticated classification tasks (beyond MNIST and CIFAR-10 image classification—such as has been shown with conventional binary neural networks [249, 281, 250]), and generative or other probabilistic tasks—for which the stochasticity can be harnessed rather than merely tolerated. Beyond machine-learning tasks, an SPDNN layer could be used as the core of a single-photon-regime photonic Ising machine [282] for heuristically solving combinatorial-optimization problems, realizing an optical version of p-bit computing [231].

9.3 Fast All-Physical Computing Systems

A practical limitation of our experiments introduced in Chapters 6 and 7 is that they were conducted with a relatively slow[1] single-photon-detector array, limiting the speed at which a single execution of a layer could be carried out, and the detector array was not optimized for energy efficiency. For our fundamen-

[1]Maximum frame rate ~ 10 kHz, mostly restricted by the readout time.

tal approach and methods to be applied to make ONNs that offer a practical advantage over state-of-the-art electronic processors as generic neural-network accelerators, there remains important work to be done in engineering an overall system that operates sufficiently fast while minimizing total energy cost. Recent progress in the development of large, fast arrays of single-photon detectors coupled with digital logic [283] suggest that there is a path towards this goal. Ref. [284] has also pointed out the possibility of using fast superconducting-nanowire single-photon detectors for realizing spiking neural networks.

Apart from trying to increase the speed of the electronic devices, a possible alternative is to construct a device that performs SPD with high efficiency and gives the measurement result as an *optical* signal that can be directly used as an input to the next layer in the ONN. Designing and demonstrating such a device is an interesting potential avenue for future work in applied quantum nonlinear optics [258, 285, 286, 287, 288], and could lead to both lower electronic energy consumption and higher speed for single-photon-detection ONNs.

9.4 Hardware-Software Co-Optimization of Physical Systems

We trained our demonstration SPDNN *in silico* using backpropagation, but if SPDNNs with high overall energy efficiency are built, it would be a boon use this efficient hardware not only for inference but also for training. To this end, it could be interesting to study how to adapt *in situ* training [289, 290, 22, 291], including backpropagation-free (e.g., Refs. [292, 293, 294, 295]), methods for SPDNNs.

While there are many reasons computer science has traditionally favored

the abstraction of hardware from software, our work is part of a broad trend, spanning many different physical platforms [9, 296, 297], in which researchers engineer computations in a physics-aware manner. By short-circuiting the abstraction hierarchy—in our case, going from a physics-aware software description of a stochastic neural network directly to a physical optical realization of the constituent operations—it is possible to achieve orders-of-magnitude improvements in energy efficiency [15, 2] versus conventional CMOS computing. *Physics-aware software*, in which software directly incorporates knowledge of the physics of the underlying computing hardware—such as in the *physics-aware probabilistic modeling* we used in this work—is understudied compared to purely software-level or hardware-level innovations (i.e., "at the top" or "at the bottom" of the hierarchy [298]). It is thus ripe for exploration: within the domain of neural networks, there are a multitude of emerging physical platforms that could be more fully harnessed if the physical devices were not forced to conform to the standard abstractions in modern computer architecture [41]. Beyond neural-network accelerators, communities such as computational imaging [299] have embraced the opportunity to improve system performance through co-optimizing hardware and software in a physics-aware manner. We believe there is an opportunity to make gains in even more areas and applications of computing technology by collapsing abstractions and implementing physics-aware software with physical hardware that could be orders of magnitude faster or more energy efficient than current digital CMOS approaches but that doesn't admit a clean, digital, deterministic abstraction.

9.5 Optical Quantum Sensing Using Optimized Systems

While the low-light sensing applications discussed in Chapter 8 benefit from optimized light-field distribution through neural-network model optimization, the entire optical system remains classical (or, quantum in a "Planck sense"). This optimization allows for higher sensitivity with a restricted photon budget for a given classification task. However, being a classical system, it cannot surpass the fundamental limits of classical light. Specifically, the SNR for detecting classical light is constrained by the square root of the detected optical energy (Figure 7.1b).

However, this shot-noise limit, or standard quantum limit in the context of quantum sensing, can be surpassed using non-classical light, such as squeezed light [300]. The field of optical quantum sensing has long utilized exotic quantum optical states to enhance sensing applications [301]. It has been shown that, with the aid of machine learning algorithms, optical quantum sensing can be extended to broader applications and achieve higher performance [302, 303].

In these applications, achieving good performance requires a complex multimode light state and an effective algorithm to optimize the system's operation [304]. The work introduced in Chapters 6 and 7 perfectly meets these requirements. By using the highly-entangled multimode states along with efficient operations proposed in Chapter 6, we can apply physically-aware probabilistic modeling of the quantum optical system to maximize performance. This procedure is similar to what we did in Chapter 7 for classical systems, except that the probability function governed by the physical process now comes from a quantum system.

9.6 Quantum Machine Learning Using Highly-Correlated Quantum Optical States

Apart from image sensing applications, the complex correlations in quantum states can also benefit machine learning tasks in general. It has been demonstrated that complex quantum states with high correlations can enhance machine learning tasks such as generative tasks [305, 306]. Utilizing the highly-multimode quantum states discussed in Chapter 6, which possess significant correlations (Figure 6.9), we can foresee that a quantum-enhanced machine learning application is possible.

Additionally, in Chapter 7, we explored probabilistic modeling of highly-stochastic samples, which closely resemble the highly-stochastic quantum samples obtained from multimode photon-counting (Figure 6.6). These samples, despite their high stochasticity, exhibit substantial correlations and complexity [146, 307]. Learning from these models could inform and enhance our approach to handling quantum samples in machine learning tasks.

APPENDIX A

MODELING OF SPDNNS

In this Appendix, we introduce the details of simulation of the single-photon-detection neural networks (SPDNNs). Each neuron activation in an SPDNN, corresponding to a readout on a single-photon detector (SPD) in an experiment, is modeled as a binary stochastic process [220, 227, 237]. For each SPD measurement, the single-shot output is either 0 or 1, with probabilities determined by the incident optical energy. The exact form of the activation function is defined by the actual physical process of single-photon detection. For an incident beam with optical energy of λ photons per detection, due to Poissonian photon statistics, the probability for an SPD to detect a click is $P_{\mathrm{SPD}}(\lambda) = 1 - e^{-\lambda}$, as shown in Figure 7.2. The detected binary results are used to compute the activation values. However, due to the stochasticity and discretization in the single-photon-detection process, estimating the gradients in the loss function is challenging, and conventional backpropagation algorithms fail to train these models.

Training stochastic neuron models has been investigated for many years. One of the major families of algorithms dependent on such neurons is the Boltzmann machine [308, 219]. REINFORCE algorithms (RA) [309] update the weights in the direction of the gradients of expected reinforcement without explicitly computing them. These algorithms have been investigated and applied in different tasks to train stochastic neural networks effectively [310, 311]. In [222], many methods of estimating gradients through stochastic neurons are studied. They found that the fastest training in their experiments was achieved by the "straight-through estimator" (STE), which was previously introduced in

Hinton's course, lecture 15b [312]. In our simulation of SPDNNs, we were inspired by both methods and found an estimator that trained our SPDNNs effectively, with the activation induced by the physical single-photon detection process. When using STE in a binary neural network, the binarization process, either deterministic or stochastic, is regarded as identity function during back propagation. However, if we directly use the STE to go "straight through" the entire SPD process, the training performance is not very good. This is because the STE is a biased estimator of the gradients [222, 313], meaning the expectation value of the estimator is not the same as the true expectation value of the real random variable. The biased estimation of gradients harms training accuracy over a large number of epochs. Then we looked for an unbiased estimator inspired by RA [309, 222]. We can conceptually break the single-photon detection process into two parts, a determinisic probability function P_{SPD}, and the Bernoulli sampling that introduces the stochasticity. For a Bernoulli distribution, the expectation value is the probability of 1 itself, so P_{SPD} is also the expectation value of the activation. Instead of going "straight through" the entire SPD process, we only skip the Bernoulli sampling process to avoid the uncertainty in backpropagation and include the gradients induced by the probability function to meet the expectation values of the random variable.

To enhance training effectiveness in certain cases, we introduced a slope variable, η, which modifies the intensity value within the SPD activation function: $P^{\eta}_{\mathrm{SPD}}(\lambda) = P_{\mathrm{SPD}}(\eta\lambda)$. The incorporation of a technique called "slope annealing" [314] allows controlled alteration of the gradients of the activation function, leading to more efficient navigation of the model's parameter space. Additionally, we impose an upper limit on the intensity by clamping it to a maximum value λ_{max}. This prevents the occurrence of vanishing gradients resulting from

excessively large values and the plateauing probability function. Both the application of the slope variable and intensity clamping can be utilized exclusively during the training phase. In optical implementation, the annealing factor can be absorbed in the mapping from the trained weights to the controlled parameters on the experimental setup.

The details of the backpropagation and training process are shown in Algorithms 1 and 3, with the exact activation functions of incoherent and coherent optical setups, respectively. In the following sections, we introduce the two SPDNN setups in detail and test their performance with different tasks and architectures.

A.1 SPDNNs with Incoherent Optical Setups

When an optical neural network (ONN) operates with incoherent light, the values of the vector elements are encoded as the intensity of light. The encoded values are non-negative, and the operations are performed by modulating the intensity of light. Thus, for an optical matrix-vector multiplier (optical MVM) operating with incoherent light, the values in an output vector z are readily the intensity to be measured by the detector, i.e., $\lambda = z$. The probability of having the SPD measurement of 1 is then $P_{\text{SPD}}(\lambda(z)) = P_{\text{SPD}}(z)$. This probability P_{SPD} is determined by the pre-activation value z. Thus, the SPD activation is a Bernoulli sampling of the probability P_{SPD}, $f_{\text{SPD}}^{\text{Incoh}}(z) = \mathbf{1}_{t<P_{\text{SPD}}(z)}$, where t is a uniform random variable $t \sim U[0,1]$ and $\mathbf{1}_x$ is the indicator function on the true value of x,

i.e.

$$f_{\mathrm{SPD}}^{\mathrm{Incoh}}(z) = \begin{cases} 1 & \text{with probability } p = P_{\mathrm{SPD}}(z), \\ 0 & \text{with probability } 1 - p, \end{cases} \tag{A.1}$$

where the probability function $P_{\mathrm{SPD}}(z) = 1 - e^{-z}$. The activation in the forward propagation is calculated as $a = f_{\mathrm{SPD}}^{\mathrm{Incoh}}(z)$.

In the backward propagation, the stochastic Bernoulli sampling process is regarded as an identity function ("straight through"), so that the gradients can propagate through the whole model as it were deterministic. In this way, the SPDNNs with stochastic binary neurons can be efficiently trained. The training procedure of incoherent SPDNNs is detailed in Algorithm 1. With L layers in the neural network, the SPD activation function is applied after every layer except the output layer. In the lth layer ($l \neq L$), $z^{(l)} = a^{(l-1)}W^{(l)}$ is the direct output of the optical MVM that encodes the information of the dot product results. In an incoherent optical setup, the output values are directly encoded in light intensity, $\lambda^{(l)} = z^{(l)}$. In the training process, we clamp the intensity to a maximum value $\lambda_{\max}$ to avoid vanishing gradients with large values. Meanwhile, the clamped intensity vector $\lambda^{(l)}$ is multiplied by the slope variable η to compute the probability of detecting a click on the SPDs according to P_{SPD}: $p^{(l)} = P_{\mathrm{SPD}}(\eta\lambda^{(l)})$. Then the activation values are the Bernoulli sampling of the computed probabilities: $a^{(l)} = \mathbf{1}_{t<p^{(l)}}$, which are sent to the next layer in the forward propagation. In backward propagation, our gradient estimator assumes the gradients of the stochastic sampling process are 1: $\partial a^{(l)}/\partial p^{(l)} = 1$. Thus, during the backward pass of the lth layer, given the gradient with respect to $a^{(l)}$, $g_{a^{(l)}} = \partial C/\partial a^{(l)}$, calculated from the next layer (previous layer in backward propagation), the gradient

with respect to the pre-activation $z^{(l)}$ is calculated as follows:

$$g_{z^{(l)}} = \frac{\partial a^{(l)}}{\partial z^{(l)}} \circ g_{a^{(l)}} = \frac{\partial a^{(l)}}{\partial p^{(l)}} \circ \frac{\partial p^{(l)}}{\partial \lambda^{(l)}} \circ \frac{\partial \lambda^{(l)}}{\partial z^{(l)}} \circ g_{a^{(l)}} = 1 \circ P'_{\mathrm{SPD}}(\lambda^{(l)}) \circ 1 \circ g_{a^{(l)}} = P'_{\mathrm{SPD}}(z^{(l)}) \circ g_{a^{(l)}}, \tag{A.2}$$

so the gradients with respect to the weights $W^{(l)}$ are $g_{W^{(l)}} = g_{z^{(l)}}^{\top} a^{(l-1)}$. In this way, the gradients can be efficiently calculated to optimize the weights using a gradient-based optimizer with a learning rate.

Note that for an incoherent optical setup, the elements in the weights (realized by intensity modulations) are also non-negative, so the updated weights need to be clamped to non-negative values after each optimization step. After each optimization step, the slope variable is updated by multiplying by a factor θ, as the "slope annealing" trick [314] to improve the training performance when necessary.

During the inference of a trained model, the forward pass of test inputs is similar to the training process, except that the maximum clamping $\lambda_{\max}$ is not applied. Additionally, to control the level of uncertainty in the stochastic neural networks, we can choose to use multiple shots of SPD measurements during each inference. In a "K-shot" inference, we use K shots of binary SPD readouts, and the final activation value of the neuron, denoted as $a^{[K]}$, is the average of the K independent stochastic binary values. This process essentially involves integrating a few more photons using the SPD, as is usually done for conventional ONN implementation [21].

Algorithm 1: Physics-aware probabilistic modeling of an SPDNN with an incoherent optical setup. N_{batch} is the batch size, N_l denotes the number of neurons in layer l and N_0 is the input size. C is the loss function. L is the number of layers. $P_{\text{SPD}}(\lambda)$ is the function of the probability to detect a click on the single-photon detector (SPD) with respect to the incident light intensity λ (in number of photons). Sample() is a probabilistic sampling function. In SPDNNs, it refers to Bernoulli sampling, Sample(p) has a probability of p to be 1 and a probability of $1-p$ to be 0 (i.e. Sample(p) $\equiv \mathbf{1}_{t<p}$, $t \sim U[0,1]$). In experiments, an SPD detection intrinsically consists both of the process, the SPD activation function $f_{\text{SPD}}(\lambda) = \mathbf{1}_{t<P_{\text{SPD}}(\lambda)}$, $t \sim U[0,1]$. The λ is equivalent to the pre-activation z in an incoherent setup. Output() determines the function applied to the pre-activation right before the final output, such as Softmax or LogSoftmax. Update() specifies how to update the parameters given the calculated gradients, using optimizers such as SGD [239], Adam [315] or AdamW [240].

Require: A batch of inputs $a^{(0)}$ ($N_{\text{batch}} \times N_0$) with corresponding targets y, current weights $W^{(l)}$ ($N_l \times N_{l-1}$, $l \in \{0, 1, \ldots, L\}$), current slope variable η, slope annealing factor θ, current learning rate α, decay coefficient γ and the clamped photon number $\lambda_{\max}$.

Ensure: Updated weights $W^{(l)}$ ($l \in \{0, 1, \ldots, L\}$), slope η and learning rate α .

I. Forward pass

for $l = 1$ to L **do**

  $z^{(l)} \leftarrow a^{(l-1)} W^{(l)\top}$ ▷ Linear operation to compute the pre-activation values

  $\lambda^{(l)} \leftarrow z^{(l)}$ ▷ For incoherent light, intensity is directly modulated

  $\lambda^{(l)} \leftarrow \min(\lambda^{(l)}, \lambda_{\max})$ ▷ Clamp the maximum intensity

  if $l < L$ **then**

    $p^{(l)} \leftarrow P_{\text{SPD}}(\eta \cdot \lambda^{(l)})$ ▷ The probability of detecting a click, with the slope η applied

    $a^{(l)} \leftarrow \text{Sample}(p^{(l)})$ ▷ SPD activation values

  end if

end for

$a^{(L)} \leftarrow \text{Output}(\lambda^{(L)})$ ▷ Final output function

II. Backward pass

Compute $g_{a^{(L)}} = \frac{\partial C}{\partial a^{(L)}}$ knowing $a^{(L)}$ and y.

$g_{z^{(L)}} \leftarrow \frac{\partial a^{(L)}}{\partial z^{(L)}} \circ g_{a^{(L)}}$

for $l = L$ to 1 **do**

  if $l < L$ **then**

    $g_{p^{(l)}} \leftarrow g_{a^{(l)}}$ ▷ "Straight-through" here, skip the Bernoulli process

    $g_{z^{(l)}} \leftarrow P'_{\text{SPD}}(z^{(l)}) \circ g_{p^{(l)}}$ ▷ $\frac{\partial p^{(l)}}{\partial z^{(l)}} = \frac{\partial p^{(l)}}{\partial \lambda^{(l)}} \circ \frac{\partial \lambda^{(l)}}{\partial z^{(l)}} = P'_{\text{SPD}}(\lambda^{(l)}) \circ 1 = P'_{\text{SPD}}(z^{(l)})$

  end if

  $g_{a^{(l-1)}} \leftarrow g_{z^{(l)}} W^{(l)}$

  $g_{W^{(l)}} \leftarrow g_{z^{(l)}}^{\top} a^{(l-1)}$ ▷ The gradients for $W^{(l)}$

end for

III. Parameter update

for $l = 1$ to L **do**

  $W^{(l)} \leftarrow \text{Update}(W^{(l)}, g_{W^{(l)}}, \alpha)$ ▷ Update the weights

  $W^{(l)} \leftarrow \max(W^{(l)}, 0)$ ▷ Clip the weights to be non-negative for the incoherent setup

end for

$\eta \leftarrow \theta\eta$ ▷ Update the slope

$\alpha \leftarrow \gamma\alpha$ ▷ Update the learning rate

For a single shot of SPD measurement per activation, $a^{[1]} = a \in \{0, 1\}$, while for K shots, $a^{[K]} = \frac{1}{K}\sum_{k=1}^{K} a_k \in \{0, 1/K, 2/K, \ldots, 1\}$. This approach reduces the uncertainty in the models, resulting in more precise output values. In the ideal case where an infinite number of shots are integrated ($K \to \infty$), the activation $a^{[\infty]}$ would converge to the expectation value without stochasticity, denoted as $a^{[\infty]} = \mathbb{E}[a] = P_{\text{SPD}}(z)$. As we will see in 7.3.1, the SPDNN models have higher test accuracy as the shots per activation K increases. The detailed inference procedure is explained in Algorithm 2.

Algorithm 2: Inference of an SPDNN with an incoherent optical setup. L is the number of layers. N_{batch} is the batch size, N_l denotes the number of neurons in layer l and N_0 is the input size. K is the number of shots used in one inference. Sample() is a probabilistic sampling function. The predictions of an inference is based on the label of the output node with the maximum output value.

Require: A batch of test inputs $a^{(0)}$ ($N_{\text{batch}} \times N_0$) and trained weights $W^{(l)}$ ($N_l \times N_{l-1}$, $l \in \{0, 1, \ldots, L\}$), slope annealing factor η.
Ensure: The output $a^{(L)}$.

for $l = 1$ to L **do**
  $z^{(l)} \leftarrow a^{(l-1)} W^{(l)\top}$ ▷ Linear operation to compute the pre-activation
  $\lambda^{(l)} \leftarrow z^{(l)}$ ▷ For incoherent light, intensity is directly modulated
  if $l < L$ **then** ▷ SPD activation process
    $p^{(l)} \leftarrow P_{\text{SPD}}(\eta \cdot \lambda^{(l)})$ ▷ The probability of detecting a click, with the slope η applied
    for $k = 1$ to K **do** ▷ K shots in one inference
      $a^{(l),k} \leftarrow \text{Sampling}(p^{(l)})$ ▷ SPD output for each shot
    end for
    $a^{(l)} \leftarrow \frac{1}{K}\sum_{k=1}^{K} a^{(l),k}$ ▷ Average over all K shots for the activation values
  end if
end for
$a^{(L)} \leftarrow \lambda^{(L)}$ ▷ Use the output intensity directly in the inference

---

A.2 SPDNNs with Coherent Optical Setups

In coherent optical MVMs [12, 63, 52, 13, 56, 17, 16], the information is conveyed through both the amplitude and phase of light states. These multipliers have the potential to encode complex numbers using arbitrary phase, but in most applications, only phases of 0 and π are used for positive and negative real-number values, to align with conventional machine learning models. Our work focuses on real-valued coherent optical MVMs. Now that the information is encoded in the amplitude and phase instead of the intensity, the photon detection process involves measuring the square modulus of the complex number, which adds an extra square function to the pre-activation values. Thus, the coherent SPD activation function is $f^{\text{Coh}}_{\text{SPD}}(z) = \mathbf{1}_{t<P_{\text{SPD}}(z^2)}$, where t is a uniform random variable $t \sim U[0,1]$ and $\mathbf{1}_x$ is the indicator function on the true value of x, i.e.

$$f^{\text{Coh}}_{\text{SPD}}(z) = \begin{cases} 1 & \text{with probability } p = P_{\text{SPD}}(z^2), \\ 0 & \text{with probability } 1-p, \end{cases} \tag{A.3}$$

where $P_{\text{SPD}}(z^2) = 1 - e^{-z^2}$. The activation in the forward propagation is calculated by $a = f^{\text{Coh}}_{\text{SPD}}(z)$. The expectation of the coherent SPD activation is $\mathbb{E}[f^{\text{Coh}}_{\text{SPD}}] = P_{\text{SPD}}(z^2)$.

The coherent activation function, depicted in Figure 7.6a, exhibits a distinct "V" shape due to the additional square operation, which is symmetric about the y axis. It could be problematic as an activation function [316]. One possible solution is to modify the information encoding and detection scheme to alter the exact form of $\lambda(z)$ (e.g. [56]). However, in this section, we have chosen to employ the most straightforward intensity-detection scenario, which does not

Algorithm 3: Physics-aware probabilistic modeling of an SPDNN with coherent light. N_{batch} is the batch size, N_l denotes the number of neurons in layer l and N_0 is the input size. C is the loss function. L is the number of layers. $P_{\text{SPD}}(\lambda)$ is the function of the probability to detect a click on the single-photon detector (SPD) with respect to the incident light intensity λ (in number of photons). Sample() is a probabilistic sampling of the probability. In SPDNNs, it refers to Bernoulli sampling, Sample(p) has a probability of p to be 1 and a probability of $1-p$ to be 0 (i.e. Sample(p) $\equiv \mathbf{1}_{t<p}$, $t \sim U[0,1]$). For a coherent setup, $\lambda = z^2$ where z is the pre-activation, output of a matrix-vector multiplier. Output() determines the function applied to the pre-activation right before the final output, such as Softmax or LogSoftmax. Update() specifies how to update the parameters given the calculated gradients, using optimizers such as SGD [239], Adam [315] or AdamW [240].

Require: A batch of inputs $a^{(0)}$ ($N_{\text{batch}} \times N_0$) with corresponding targets y, current weights $W^{(l)}$ ($N_l \times N_{l-1}$, $l \in \{0, 1, \ldots, L\}$), current slope variable η, slope annealing factor θ, current learning rate α, decay coefficient γ and the clamped photon number $\lambda_{\max}$.

Ensure: Updated weights $W^{(l)}$ ($l \in \{0, 1, \ldots, L\}$), slope η and learning rate α.

I. Forward pass

for $l = 1$ to L **do**

  $z^{(l)} \leftarrow a^{(l-1)} W^{(l)\top}$ ▷ Linear operation to compute the pre-activation

  $\lambda^{(l)} \leftarrow (z^{(l)})^2$ ▷ For coherent light, intensity is the square of the amplitude

  $\lambda^{(l)} \leftarrow \min(\lambda^{(l)}, \lambda_{\max})$ ▷ Clamp the maximum intensity

  if $l < L$ **then**

    $p^{(l)} \leftarrow P_{\text{SPD}}(\lambda^{(l)})$ ▷ The probability of detecting a click on the SPDs

    $a^{(l)} \leftarrow \text{Sample}(p^{(l)})$ ▷ SPD output for each shot

  end if

end for

$a^{(L)} \leftarrow \text{Output}(\lambda^{(L)})$ ▷ Final output function

II. Backward pass

Compute $g_{a^{(L)}} = \frac{\partial C}{\partial a^{(L)}}$ knowing $a^{(L)}$ and y.

$g_{z^{(L)}} \leftarrow \frac{\partial a^{(L)}}{\partial z^{(L)}} \circ g_{a^{(L)}}$

for $l = L$ to 1 **do**

  if $l < L$ **then**

    $g_{p^{(l)}} \leftarrow g_{a^{(l)}}$ ▷ "Straight-through" here, skip the Bernoulli process

    $g_{z^{(l)}} \leftarrow 2z^{(l)} \circ P'_{\text{SPD}}\left((z^{(l)})^2\right) \circ g_{p^{(l)}}$ ▷ $\frac{\partial p^{(l)}}{\partial z^{(l)}} = \frac{\partial p^{(l)}}{\partial \lambda^{(l)}} \circ \frac{\partial \lambda^{(l)}}{\partial z^{(l)}} = 2z^{(l)} \circ P'_{\text{SPD}}\left((z^{(l)})^2\right)$

  end if

  $g_{a^{(l-1)}} \leftarrow g_{z^{(l)}} W^{(l)}$

  $g_{W^{(l)}} \leftarrow g_{z^{(l)}}^{\top} a^{(l-1)}$ ▷ The gradients with respect to $W^{(l)}$

end for

III. Parameter update

for $l = 1$ to L **do**

  $W^{(l)} \leftarrow \text{Update}(W^{(l)}, g_{W^{(l)}}, \alpha)$ ▷ Update the weights

end for

$\eta \leftarrow \theta\eta$ ▷ Update the slope

$\alpha \leftarrow \gamma\alpha$ ▷ Update the learning rate

necessitate modifications to conventional ONN implementation. Remarkably, despite its simplicity, this activation function delivers comparable performance and demonstrates impressive results. By adopting this approach, we alleviate experimental complexities while ensuring reliable inference in our SPDNN models.

APPENDIX B

EXAMPLE PYTHON CODE

B.1 Source Code for SPD Activation Functions

```
import torch
import torch.nn as nn
from torch.autograd import Function

class PhotonCountingP(nn.Module):
    """ The probability of 1 photon in photon counting
        (also the expectation value) with mean flux x """
    def __init__(self):
        super(PhotonCountingP, self).__init__()

    def forward(self, x):
        return 1.-torch.exp(torch.abs(x)*-1.)

class BernoulliFunctionST(Function):
    """ The 'Straight Through' stochastic Bernoulli activation"""
    @staticmethod
    def forward(ctx, input):

        return torch.bernoulli(input)

    @staticmethod
    def backward(ctx, grad_output):

        return grad_output

class PoissonFunctionST(Function):
```

```
    """ The 'Straight Through' stochastic Poisson activation"""
    @staticmethod
    def forward(ctx, input):

        return torch.poisson(input)

    @staticmethod
    def backward(ctx, grad_output):

        return grad_output

PoissonST = PoissonFunctionST.apply
BernoulliST = BernoulliFunctionST.apply

class PhotonActivation(nn.Module):
    """ Single-photon-detection activation function taking light
   intensity as input """
    def __init__(self,sampler='bernoulli'):
        super(PhotonActivation, self).__init__()
        self.act = PhotonCountingP()
        if sampler == 'poisson':  # Photon-number-resolving detection
            self.sampler = PoissonST
        elif sampler == 'bernoulli':  # Single-photon detection
            self.sampler = BernoulliST
        else:
            raise

    def forward(self, input, n_rep=1, slope=1.):
        x = input
        probs = self.act(slope * x)
        out = self.sampler(probs)
        if self.sampler == BernoulliST:
```

```
            probs = self.act(x)
        elif self.sampler == PoissonST:
            probs = torch.abs(x)
        else: raise
        if n_rep==0:  # Infinite number of shots per activation
            out = probs
        else:
            out = self.sampler(probs.unsqueeze(0).repeat((n_rep,)
    +(1,)*len(probs.shape))).mean(axis=0)*torch.sign(x)
        return out
        out = self.sampler(probs)
        return out

class PhotonActivationCoh(nn.Module):
    """ Single-photon-detection activation function taking coherent
    amplitude as input """
    def __init__(self,sampler='bernoulli'):
        super(PhotonActivationCoh, self).__init__()
        self.act = PhotonCountingP()
        if sampler == 'poisson':  # Photon-number-resolving detection
            self.sampler = PoissonST
        elif sampler == 'bernoulli':  # Single-photon detection
            self.sampler = BernoulliST
        else:
            raise

    def forward(self, input, n_rep=1, slope=1.):
        x = input**2
        probs = self.act(slope * x)
        out = self.sampler(probs)
        if self.sampler == BernoulliST:
            probs = self.act(x)
```

```
        elif self.sampler == PoissonST:
            probs = torch.abs(x)
        else:
            raise
        if n_rep==0:  # Infinite number of shots per activation
            out = probs
        else:
            out = self.sampler(probs.unsqueeze(0).repeat((n_rep,)
    +(1,)*len(probs.shape))).mean(axis=0)*torch.sign(x)
        return out
```

Listing B.1: PhotonActivation.py

B.2 Example Code for Different SPDNN Architechture

```
import torch.nn.functional as F
import torch.nn as nn
from PhotonActivation import PhotonActivation, PhotonActivationCoh

class incoh_PDMLP(nn.Module):
    """ MLP-SPDNN models using incoherent light """
    def __init__(self, n_hiddens=[100,100], n_input=784, n_output=10,
     sampler='bernoulli',output_bias=True):
        super(incoh_PDMLP, self).__init__()

        self.sampler = sampler

        self.n_input = n_input
        self.n_output = n_output
        self.n_hiddens = n_hiddens

```

```
        n_nodes = [n_input]+list(n_hiddens)
        self.fcs = nn.ModuleList([nn.Linear(i,j,bias=False) for i, j
    in zip(n_nodes[:-1], n_nodes[1:])])
        self.last_fc = nn.Linear(n_hiddens[-1],n_output,bias=
    output_bias)
        self.act = PhotonActivation(sampler=sampler)

    def forward(self, x, n_rep=1, slope=1.):
        x = x.view(-1, self.n_input)
        for fc in self.fcs:
            x = fc(x)
            x = self.act(x, n_rep=n_rep, slope=slope)
        x_out = F.log_softmax(self.last_fc(x), dim=1)
        return x_out

class coh_PDMLP(nn.Module):
    """ MLP-SPDNN models using coherent light """
    def __init__(self, n_hiddens=[100,100], n_input=784, n_output=10,
     sampler='bernoulli',output_bias=True):
        super(coh_PDMLP, self).__init__()

        self.sampler = sampler

        self.n_input = n_input
        self.n_output = n_output
        self.n_hiddens = n_hiddens

        n_nodes = [n_input]+list(n_hiddens)
        self.fcs = nn.ModuleList([nn.Linear(i,j,bias=False) for i, j
    in zip(n_nodes[:-1], n_nodes[1:])])
        self.last_fc = nn.Linear(n_hiddens[-1],n_output,bias=
    output_bias)
```

```
        self.act = PhotonActivationCoh(sampler=sampler)

    def forward(self, x, n_rep=1, slope=1.):
        x = x.view(-1, self.n_input)
        for fc in self.fcs:
            x = fc(x)
            x = self.act(x, n_rep=n_rep, slope=slope)
        x_out = F.log_softmax(self.last_fc(x), dim=1)
        return x_out
```

Listing B.2: SPDNN_MLP.py

```
import torch.nn.functional as F
import torch.nn as nn
from PhotonActivation import PhotonActivationCoh
import numpy as np

def PDConv(n_in=128, n_out=128, s=1, ks=3, batchnorm=True):
    if batchnorm:
        return [
            nn.Conv2d(in_channels=n_in, out_channels=n_out,
   kernel_size=ks, stride=s, padding=int((ks-1)/2*s),bias=False),
            nn.BatchNorm2d(n_out),
            PhotonActivationCoh()
        ]
    else:
        return [
            nn.Conv2d(in_channels=n_in, out_channels=n_out,
   kernel_size=ks, stride=s, padding=int((ks-1)/2*s),bias=False),
            PhotonActivationCoh()
        ]
```

```
18
19 def PDConvsAP(n_in=3, n_chan=[128,128], ss=[1,1], kss=[3,3],
       batchnorm=True):
20     modules = []
21     n_list = [n_in]+list(n_chan)
22     for i in range(len(n_chan)):
23         modules += PDConv(n_in=n_list[i],n_out=n_list[i+1],s=ss[i],ks
       =kss[i],batchnorm=batchnorm)\
24                         +[nn.AvgPool2d((2,2))]
25
26     return nn.Sequential(*modules)
27
28 class PDConvNet(nn.Module):
29
30     def __init__(self, n_linear=100, n_output=10, d_input=(1,28,28),
       n_chan=[128,128], ss=[1,1], kss=[3,3], batchnorm=True, dropout=
       None, last_layer_bias=True, linear_act='PD', sampler='bernoulli'):
31         super(PDConvNet, self).__init__()
32
33         self.sampler = sampler
34         self.n_chan = n_chan
35         self.batchnorm = batchnorm
36
37         self.d_input = d_input
38         self.n_output = n_output
39         self.n_linear = n_linear
40
41         self.convs = PDConvsAP(n_in=d_input[0],n_chan=n_chan,ss=ss,
       kss=kss,batchnorm=batchnorm)
42         self.flat = nn.Flatten()
43
44         self.fc1 = nn.Linear(int(n_chan[-1]*(d_input[1]//2**len(
```

```
    n_chan)//np.prod(ss))**2), n_linear, bias=False)
45          if linear_act=='PD':
46              self.linear_act = PhotonActivationCoh(sampler=sampler)
47          elif linear_act=='ReLU':
48              self.linear_act = nn.ReLU()
49          self.fc2 = nn.Linear(n_linear, n_output, bias=last_layer_bias
    )
50          if dropout is None:
51              self.dropout = None
52          else:
53              self.dropout = nn.Dropout(dropout)
54
55      def forward(self, x, n_rep=1, slope=1.):
56          x = x.view(-1, *self.d_input)
57          for layer in self.convs:
58              if isinstance(layer, PhotonActivationCoh):
59                  x = layer(x, n_rep=n_rep, slope=slope)
60              else:
61                  x = layer(x)
62          x = self.flat(x)
63          x = self.fc1(x)
64          if isinstance(self.linear_act, PhotonActivationCoh):
65              x = self.linear_act(x, n_rep=n_rep, slope=slope)
66          else:
67              x = self.linear_act(x)
68          if self.dropout is not None:
69              x = self.dropout(x)
70          x = self.fc2(x)
71          x_out = F.log_softmax(x, dim=1)
72          return x_out
```

Listing B.3: SPDNN_Conv.py

BIBLIOGRAPHY

[1] T. Wang, S.-Y. Ma, L. G. Wright, T. Onodera, B. C. Richard, and P. L. McMahon, An optical neural network using less than 1 photon per multiplication. *Nature Communications* **13**, 1–8 (2022).

[2] M. Anderson, S.-Y. Ma, T. Wang, L. Wright, and P. McMahon, Optical Transformers. *Transactions on Machine Learning Research* (2024).

[3] F. Presutti, L. G. Wright, S.-Y. Ma, T. Wang, B. K. Malia, T. Onodera, and P. L. McMahon, Highly multimode visible squeezed light with programmable spectral correlations through broadband up-conversion. *arXiv:2401.06119* (2024).

[4] S.-Y. Ma, T. Wang, J. Laydevant, L. G. Wright, and P. L. McMahon, Quantum-noise-limited optical neural networks operating at a few quanta per activation. *arXiv:2307.15712* (2023).

[5] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning. *Nature* **521**, 436–444 (2015).

[6] A. Canziani, A. Paszke, and E. Culurciello, An analysis of deep neural network models for practical applications. *arXiv:1605.07678* (2016).

[7] N. C. Thompson, K. Greenewald, K. Lee, and G. F. Manso, The computational limits of deep learning. *arXiv:2007.05558* (2020).

[8] D. Marković, A. Mizrahi, D. Querlioz, and J. Grollier, Physics for neuromorphic computing. *Nature Reviews Physics* **2**, 499–510 (2020).

[9] D. V. Christensen, R. Dittmann, B. Linares-Barranco, A. Sebastian, M. Le Gallo, A. Redaelli, S. Slesazeck, T. Mikolajick, S. Spiga, S. Menzel et al. 2022 roadmap on neuromorphic computing and engineering. *Neuromorphic Computing and Engineering* **2**, 022501 (2022).

[10] A. Jassy, *Keynote address at AWS re:Invent*. URL: `www.youtube.com/watch?v=7-31KgImGgU` (2019).

[11] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, Efficient processing of deep neural networks: A tutorial and survey. *Proceedings of the IEEE* **105**, 2295–2329 (2017).

[12] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund et al. Deep learning with coherent nanophotonic circuits. *Nature Photonics* **11**, 441 (2017).

[13] X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, and A. Ozcan, All-optical machine learning using diffractive deep neural networks. *Science* **361**, 1004–1008 (2018).

[14] C. Ríos, N. Youngblood, Z. Cheng, M. Le Gallo, W. H. Pernice, C. D. Wright, A. Sebastian, and H. Bhaskaran, In-memory computing on a photonic platform. *Science Advances* **5**, eaau5759 (2019).

[15] G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljačić, C. Denz, D. A. Miller, and D. Psaltis, Inference in artificial intelligence with deep optics and photonics. *Nature* **588**, 39–47 (2020).

[16] X. Xu, M. Tan, B. Corcoran, J. Wu, A. Boes, T. G. Nguyen, S. T. Chu, B. E. Little, D. G. Hicks, R. Morandotti, A. Mitchell, and D. J. Moss, 11 TOPS photonic convolutional accelerator for optical neural networks. *Nature* **589**, 44–51 (2021).

[17] J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. S. Raja et al. Parallel convolutional processing using an integrated photonic tensor core. *Nature* **589**, 52–58 (2021).

[18] T. Zhou, X. Lin, J. Wu, Y. Chen, H. Xie, Y. Li, J. Fan, H. Wu, L. Fang, and Q. Dai, Large-scale neuromorphic optoelectronic computing with a reconfigurable diffractive processing unit. *Nature Photonics* **15**, 367–373 (2021).

[19] R. Davis III, Z. Chen, R. Hamerly, and D. Englund, Frequency-encoded deep learning with speed-of-light dominated latency. *arXiv:2207.06883* (2022).

[20] F. Ashtiani, A. J. Geers, and F. Aflatouni, An on-chip photonic deep neural network for image classification. *Nature* **606**, 501–506 (2022).

[21] A. Sludds, S. Bandyopadhyay, Z. Chen, Z. Zhong, J. Cochrane, L. Bernstein, D. Bunandar, P. B. Dixon, S. A. Hamilton, M. Streshinsky et al. Delocalized photonic deep learning on the internet's edge. *Science* **378**, 270–276 (2022).

[22] S. Bandyopadhyay, A. Sludds, S. Krastanov, R. Hamerly, N. Harris, D. Bunandar, M. Streshinsky, M. Hochberg, and D. Englund, Single chip photonic deep neural network with accelerated training. *arXiv:2208.01623* (2022).

[23] B. J. Shastri, A. N. Tait, T. F. de Lima, W. H. Pernice, H. Bhaskaran, C. D. Wright, and P. R. Prucnal, Photonics for artificial intelligence and neuromorphic computing. *Nature Photonics* **15**, 102–114 (2021).

[24] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X* **9**, 021032 (2019).

[25] M. A. Nahmias, T. F. De Lima, A. N. Tait, H.-T. Peng, B. J. Shastri, and P. R. Prucnal, Photonic multiply-accumulate operations for neural networks. *IEEE Journal of Selected Topics in Quantum Electronics* **26**, 1–18 (2020).

[26] H. J. Caulfield and S. Dolev, Why future supercomputing requires optics. *Nature Photonics* **4**, 261–263 (2010).

[27] A. Reuther, P. Michaleas, M. Jones, V. Gadepally, S. Samsi, and J. Kepner, Survey of machine learning accelerators. *arXiv:2009.00993* (2020).

[28] Graphcore The data center architecture for graphcore computing. (2021).

[29] Cerebras Systems Cerebras systems: Achieving industry best AI performance through a systems approach. (2021).

[30] Michael Andersch and Greg Palmer and Ronny Krashinsky and Nick Stam and Vishal Mehta and Gonzalo Brito and Sridhar Ramaswamy NVIDIA Hopper architecture in-depth. (2022).

[31] Habana Labs HABANA® GAUDI®2 white paper. (2022).

[32] A. Sebastian, M. L. Gallo, R. Khaddam-Aljameh, and E. Eleftheriou, Memory devices and applications for in-memory computing. *Nature Nanotechnology* **15**, 529–544 (2020).

[33] P. Stark, F. Horst, R. Dangel, J. Weiss, and B. J. Offrein, Opportunities for integrated photonic neural networks. *Nanophotonics* **9**, 4221–4232 (2020).

[34] C. Huang, V. J. Sorger, M. Miscuglio, M. Al-Qadasi, A. Mukherjee, L. Lampe, M. Nichols, A. N. Tait, T. F. de Lima, B. A. Marquez, J. Wang, L. Chrostowski, M. P. Fok, D. Brunner, S. Fan, S. Shekhar, P. R. Prucnal, and B. J. Shastri, Prospects and applications of photonic neural networks. *Advances in Physics: X* **7** (2021).

[35] S. Moon, K. Shin, and D. Jeon, Enhancing reliability of analog neural network processors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **27**, 1455–1459 (2019).

[36] V. Joshi, M. Le Gallo, S. Haefeli, I. Boybat, S. R. Nandakumar, C. Piveteau, M. Dazzi, B. Rajendran, A. Sebastian, and E. Eleftheriou, Accurate deep neural network inference using computational phase-change memory. *Nature Communications* **11**, 1–13 (2020).

[37] N. Semenova, L. Larger, and D. Brunner, Understanding and mitigating noise in trained deep neural networks. *Neural Networks* **146**, 151–160 (2022).

[38] M. Klachko, M. R. Mahmoodi, and D. Strukov, Improving noise tolerance of mixed-signal neural networks. In *2019 International Joint Conference on Neural Networks (IJCNN)*, 1–8 (2019).

[39] C. Zhou, P. Kadambi, M. Mattina, and P. N. Whatmough, Noisy machines: Understanding noisy neural networks and enhancing robustness to analog hardware errors using distillation. *arXiv:2001.04974* (2020).

[40] X. Yang, C. Wu, M. Li, and Y. Chen, Tolerating Noise Effects in Processing-in-Memory Systems for Neural Networks: A Hardware–Software Codesign Perspective. *Advanced Intelligent Systems* **4**, 2200029 (2022).

[41] L. G. Wright, T. Onodera, M. M. Stein, T. Wang, D. T. Schachter, Z. Hu, and P. L. McMahon, Deep physical neural networks trained with backpropagation. *Nature* **601**, 549–555 (2022).

[42] N. Semenova and D. Brunner, Noise-mitigation strategies in physical feedforward neural networks. *Chaos: An Interdisciplinary Journal of Nonlinear Science* **32**, 061106 (2022).

[43] C. Wu, X. Yang, H. Yu, R. Peng, I. Takeuchi, Y. Chen, and M. Li, Harnessing optoelectronic noises in a photonic generative network. *Science Advances* **8**, eabm2956 (2022).

[44] M. G. Anderson, S.-Y. Ma, T. Wang, L. G. Wright, and P. L. McMahon, Optical transformers. *arXiv:2302.10360* (2023).

[45] Y. Jiang, W. Zhang, X. Liu, W. Zhu, J. Du, and Z. He, Physical Layer-aware Digital-Analog Co-Design for Photonic Convolution Neural Network. *IEEE Journal of Selected Topics in Quantum Electronics* (2023).

[46] L. Bernstein, A. Sludds, C. Panuski, S. Trajtenberg-Mills, R. Hamerly, and D. Englund, Single-shot optical neural network. *Science Advances* **9**, eadg7904 (2023).

[47] A. N. Tait, Quantifying power use in silicon photonic neural networks. *arXiv:2108.04819* (2021).

[48] C. Mesaritakis, V. Papataxiarhis, and D. Syvridis, Micro ring resonators as building blocks for an all-optical high-speed reservoir-computing bit-pattern-recognition system. *J. Opt. Soc. Am. B* **30**, 3048–3055 (2013).

[49] A. N. Tait, A. X. Wu, T. F. de Lima, E. Zhou, B. J. Shastri, M. A. Nahmias, and P. R. Prucnal, Microring Weight Banks. *IEEE Journal of Selected Topics in Quantum Electronics* **22**, 312–325 (2016).

[50] G. Giamougiannis, A. Tsakyridis, Y. Ma, A. Totović, M. Moralis-Pegios, D. Lazovsky, and N. Pleros, A Coherent Photonic Crossbar for Scalable Universal Linear Optics. *Journal of Lightwave Technology* **41**, 2425–2442 (2023).

[51] J. Carolan, C. Harrold, C. Sparrow, E. Martín-López, N. J. Russell, J. W. Silverstone, P. J. Shadbolt, N. Matsuda, M. Oguma, M. Itoh et al. Universal linear optics. *Science* **349**, 711–716 (2015).

[52] W. Bogaerts, D. Pérez, J. Capmany, D. A. B. Miller, J. Poon, D. Englund, F. Morichetti, and A. Melloni, Programmable photonic circuits. *Nature* **586**, 207–216 (2020).

[53] J. Gu, H. Zhu, C. Feng, Z. Jiang, R. Chen, and D. Pan, L2ight: Enabling On-Chip Learning for Optical Neural Networks via Efficient in-situ Subspace Optimization. In *Advances in Neural Information Processing Systems* eds. M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Vol. 34, 8649–8661 (2021).

[54] D. A. B. Miller, Why optics needs thickness. *Science* **379**, 41–45 (2023).

[55] Y. Hayasaki, I. Tohyama, T. Yatagai, M. Mori, and S. Ishihara, Optical learning neural network using Selfoc microlens array. *Japanese Journal of Applied Physics* **31**, 1689 (1992).

[56] J. Spall, X. Guo, T. D. Barrett, and A. Lvovsky, Fully reconfigurable coherent optical vector–matrix multiplication. *Optics Letters* **45**, 5752–5755 (2020).

[57] J. Chang, V. Sitzmann, X. Dun, W. Heidrich, and G. Wetzstein, Hybrid optical-electronic convolutional neural networks with optimized diffractive optics for image classification. *Scientific Reports* **8**, 12324 (2018).

[58] X. Meng, G. Zhang, N. Shi, G. Li, J. Azaña, J. Capmany, J. Yao, Y. Shen, W. Li, N. Zhu, and M. Li, Compact optical convolution processing unit based on multimode interference. *Nature Communications* **14** (2023).

[59] H. Zheng, Q. Liu, I. I. Kravchenko, X. Zhang, Y. Huo, and J. G. Valentine, Intelligent Multi-channel Meta-imagers for Accelerating Machine Vision. *arXiv:2306.07365* (2023).

[60] A. N. Tait, J. Chang, B. J. Shastri, M. A. Nahmias, and P. R. Prucnal, Demonstration of WDM weighted addition for principal component analysis. *Optics Express* **23**, 12758–12765 (2015).

[61] A. N. Tait, T. F. De Lima, M. A. Nahmias, H. B. Miller, H.-T. Peng, B. J. Shastri, and P. R. Prucnal, Silicon photonic modulator neuron. *Physical Review Applied* **11**, 064043 (2019).

[62] C. Wu, H. Yu, S. Lee, R. Peng, I. Takeuchi, and M. Li, Programmable phase-change metasurfaces on waveguides for multimode photonic convolutional neural network. *Nature Communications* **12**, 1–8 (2021).

[63] M. Miscuglio, Z. Hu, S. Li, J. K. George, R. Capanna, H. Dalir, P. M. Bardet, P. Gupta, and V. J. Sorger, Massively parallel amplitude-only Fourier neural network. *Optica* **7**, 1812–1819 (2020).

[64] J. W. Goodman, A. Dias, and L. Woody, Fully parallel, high-speed incoherent optical method for performing discrete Fourier transforms. *Optics Letters* **2**, 1–3 (1978).

[65] D. Psaltis, D. Brady, and K. Wagner, Adaptive optical networks using photorefractive crystals. *Applied Optics* **27**, 1752–1759 (1988).

[66] J. Dong, M. Rafayelyan, F. Krzakala, and S. Gigan, Optical reservoir computing using multiple light scattering for chaotic systems prediction. *IEEE Journal of Selected Topics in Quantum Electronics* **26**, 1–12 (2019).

[67] M. W. Matthès, P. del Hougne, J. de Rosny, G. Lerosey, and S. M. Popoff, Optical complex media as universal reconfigurable linear operators. *Optica* **6**, 465–472 (2019).

[68] J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, Reinforcement learning in a large-scale photonic recurrent neural network. *Optica* **5**, 756–760 (2018).

[69] L. Bernstein, A. Sludds, R. Hamerly, V. Sze, J. Emer, and D. Englund, Freely scalable and reconfigurable optical hardware for deep learning. *Scientific Reports* **11**, 1–12 (2021).

[70] C. Ramey, Silicon photonics for artificial intelligence acceleration. In *Proceedings of the IEEE Hot Chips 32 Symposium (HCS)*, 1–26 (2020).

[71] D. A. B. Miller, Waves, modes, communications, and optics: a tutorial. *Advances in Optics and Photonics* **11**, 679–825 (2019).

[72] I. R. Berchera and I. P. Degiovanni, Quantum imaging with sub-Poissonian light: challenges and perspectives in optical metrology. *Metrologia* **56**, 024001 (2019).

[73] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, C. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmagham et al. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture (ISCA)*, 1–12 (2017).

[74] D. A. B. Miller, Attojoule optoelectronics for low-energy information processing and communications. *Journal of Lightwave Technology* **35**, 346–396 (2017).

[75] M. Horowitz, Computing's energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, 10–14 (2014).

[76] W. Andregg, M. Andregg, R. T. Weverka, and L. Clermont, Wavelength

multiplexed matrix-matrix multiplier. (U.S. Patent No. 10,274,989). U.S. Patent and Trademark Office (2019).

[77] A. Hemmi, R. Mizumura, R. Kawanishi, H. Nakajima, H. Zeng, K. Uchiyama, N. Kaneki, and T. Imato, Development of a novel two dimensional surface plasmon resonance sensor using multiplied beam splitting optics. *Sensors* **13**, 801–812 (2013).

[78] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. *https://arxiv.org/abs/1712.05877* (2017).

[79] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019).

[80] P. De Chazal, J. Tapson, and A. Van Schaik, A comparison of extreme learning machines and back-propagation trained feed-forward networks processing the MNIST database. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2165–2168 (2015).

[81] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, How to evaluate deep neural network processors: TOPS/W (alone) considered harmful. *IEEE Solid-State Circuits Magazine* **12**, 28–41 (2020).

[82] J. Zhang, Z. Wang, and N. Verma, A machine-learning classifier implemented in a standard 6T SRAM array. In *2016 IEEE Symposium on VLSI Circuits (VLSI-Circuits)*, 1–2 (2016).

[83] B. Moons, D. Bankman, L. Yang, B. Murmann, and M. Verhelst, BinarEye: An always-on energy-accuracy-scalable binary CNN processor with all memory on chip in 28nm CMOS. In *2018 IEEE Custom Integrated Circuits Conference (CICC)*, 1–4 (2018).

[84] J. Park, J. Lee, and D. Jeon, A 65-nm neuromorphic image classification processor with energy-efficient training through direct spike-only feedback. *IEEE Journal of Solid-State Circuits* **55**, 108–119 (2019).

[85] X. Xu, M. Tan, B. Corcoran, J. Wu, T. G. Nguyen, A. Boes, S. T. Chu, B. E. Little, R. Morandotti, A. Mitchell, D. G. Hicks, and D. J. Moss, Photonic

perceptron based on a soliton crystal Kerr microcomb for high-speed, scalable, optical neural networks. *arXiv:2003.01347* (2020).

[86] Y. Su, Y. Zhang, C. Qiu, X. Guo, and L. Sun, Silicon photonic platform for passive waveguide devices: materials, fabrication, and applications. *Advanced Materials Technologies* **5**, 1901153 (2020).

[87] G. W. Burr, R. M. Shelby, S. Sidler, C. Di Nolfo, J. Jang, I. Boybat, R. S. Shenoy, P. Narayanan, K. Virwani, E. U. Giacometti, B. Kurdi, and H. Hwang, Experimental demonstration and tolerancing of a large-scale neural network (165 000 synapses) using phase-change memory as the synaptic weight element. *IEEE Transactions on Electron Devices* **62**, 3498–3507 (2015).

[88] N. Youngblood, C. Chen, S. J. Koester, and M. Li, Waveguide-integrated black phosphorus photodetector with high responsivity and low dark current. *Nature Photonics* **9**, 247–252 (2015).

[89] C. De Sa, C. Zhang, K. Olukotun, and C. Ré, Taming the wild: A unified analysis of hogwild!-style algorithms. *Advances in Neural Information Processing Systems (NeurIPS)* **28**, 2656 (2015).

[90] M. Prabhu, C. Roques-Carmes, Y. Shen, N. Harris, L. Jing, J. Carolan, R. Hamerly, T. Baehr-Jones, M. Hochberg, V. Čeperić, J. D. Joannopoulos, D. R. Englund, and M. Soljačić, Accelerating recurrent Ising machines in photonic integrated circuits. *Optica* **7**, 551–558 (2020).

[91] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara, R. L. Byer, M. M. Fejer, H. Mabuchi, and Y. Yamamoto, A fully programmable 100-spin coherent Ising machine with all-to-all connections. *Science* **354**, 614–617 (2016).

[92] T. Inagaki, Y. Haribara, K. Igarashi, T. Sonobe, S. Tamate, T. Honjo, A. Marandi, P. L. McMahon, T. Umeki, K. Enbutsu, O. Tadanaga, H. Takenouchi, K. Aihara, K.-i. Kawarabayashi, K. Inoue, S. Utsunomiya, and H. Takesue, A coherent Ising machine for 2000-node optimization problems. *Science* **354**, 603–606 (2016).

[93] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, Attention is All you Need. In *Advances in Neural Information Processing Systems*, 5998-6008 (2017).

[94] OpenAI, *ChatGPT: Optimizing Language Models for Dialogue* (`https://www.openai.com/blog/chatgpt`). (2022).

[95] T. B. Brown, et al. *Language Models are Few-Shot Learners*. (2020).

[96] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, *Scaling Laws for Neural Language Models*. (2020).

[97] A. Clark, et al. Unified Scaling Laws for Routed Language Models. In *Proceedings of the 39th International Conference on Machine Learning*, Proceedings of Machine Learning Research eds. K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Vol. 162, 4057–4086 (2022).

[98] J. Hoffmann, et al. *Training Compute-Optimal Large Language Models*. (2022).

[99] M. Treviso, T. Ji, J.-U. Lee, B. van Aken, Q. Cao, M. R. Ciosici, M. Hassid, K. Heafield, S. Hooker, P. H. Martins, A. F. T. Martins, P. Milder, C. Raffel, E. Simpson, N. Slonim, N. Balasubramanian, L. Derczynski, and R. Schwartz, *Efficient Methods for Natural Language Processing: A Survey*. (2022).

[100] X. Zhai, A. Kolesnikov, N. Houlsby, and L. Beyer, Scaling Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12104-12113 (2022).

[101] OpenAI, et al. *GPT-4 Technical Report*. (2024).

[102] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *ArXiv* **abs/1910.01108** (2019).

[103] J. Sevilla, L. Heim, A. Ho, T. Besiroglu, M. Hobbhahn, and P. Villalobos, Compute Trends Across Three Eras of Machine Learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, 1-8 (2022).

[104] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *ICLR* (2021).

[105] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, Swin

Transformer: Hierarchical Vision Transformer using Shifted Windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* (2021).

[106] J. Kim, T. D. Nguyen, S. Min, S. Cho, M. Lee, H. Lee, and S. Hong, Pure Transformers are Powerful Graph Learners. *arXiv* **abs/2207.02505** (2022).

[107] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, and J. Carreira, Perceiver: General Perception with Iterative Attention. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research eds. M. Meila and T. Zhang, Vol. 139, 4651–4664 (2021).

[108] A. Jaegle, S. Borgeaud, J.-B. Alayrac, C. Doersch, C. Ionescu, D. Ding, S. Koppula, D. Zoran, A. Brock, E. Shelhamer, O. Hénaff, M. M. Botvinick, A. Zisserman, O. Vinyals, and J. Carreira, *Perceiver IO: A General Architecture for Structured Inputs & Outputs*. (2021).

[109] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research eds. M. Meila and T. Zhang, Vol. 139, 8748–8763 (2021).

[110] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever, Zero-Shot Text-to-Image Generation. In *Proceedings of the 38th International Conference on Machine Learning*, Proceedings of Machine Learning Research eds. M. Meila and T. Zhang, Vol. 139, 8821–8831 (2021).

[111] J. Yu, Z. Wang, V. Vasudevan, L. Yeung, M. Seyedhosseini, and Y. Wu, CoCa: Contrastive Captioners are Image-Text Foundation Models. *Transactions on Machine Learning Research* (2022).

[112] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-maron, M. Giménez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas, A Generalist Agent. *Transactions on Machine Learning Research* (2022). Featured Certification.

[113] A. Radford and K. Narasimhan, Improving Language Understanding by Generative Pre-Training (2018).

[114] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota, 4171–4186 (2019).

[115] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, Language Models are Unsupervised Multitask Learners. (2019).

[116] K. Lu, A. Grover, P. Abbeel, and I. Mordatch, Pretrained Transformers as Universal Computation Engines. *arXiv preprint arXiv:2103.05247* (2021).

[117] A. Lewkowycz, A. Andreassen, D. Dohan, E. Dyer, H. Michalewski, V. Ramasesh, A. Slone, C. Anil, I. Schlag, T. Gutman-Solo, Y. Wu, B. Neyshabur, G. Gur-Ari, and V. Misra, *Solving Quantitative Reasoning Problems with Language Models*. (2022).

[118] A. Krizhevsky, I. Sutskever, and G. E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems* eds. F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Vol. 25 (2012).

[119] Z. Chen, A. Sludds, R. Davis, I. Christen, L. Bernstein, L. Ateshian, T. Heuser, N. Heermeier, J. A. Lott, S. Reitzenstein, R. Hamerly, and D. Englund, Deep learning with coherent VCSEL neural networks. *Nature Photonics* **17**, 723–730 (2023).

[120] Z. Li, E. Wallace, S. Shen, K. Lin, K. Keutzer, D. Klein, and J. E. Gonzalez, Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20 (2020).

[121] Y. Bondarenko, M. Nagel, and T. Blankevoort, Understanding and Overcoming the Challenges of Efficient Transformer Quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic, 7947–7969 (2021).

[122] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, *LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale*. (2022).

[123] T. Dettmers and L. Zettlemoyer, The case for 4-bit precision: k-bit Inference Scaling Laws. *arXiv:2212.09720* (2022).

[124] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko, Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2704–2713 (2018).

[125] K. He, X. Zhang, S. Ren, and J. Sun, Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016).

[126] J. L. Ba, J. R. Kiros, and G. E. Hinton, *Layer Normalization*. (2016).

[127] D. Hendrycks and K. Gimpel, *Gaussian Error Linear Units (GELUs)*. (2016).

[128] A. Krizhevsky, Convolutional Deep Belief Networks on CIFAR-10 (2010).

[129] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv:1704.04861* (2017).

[130] H. Kim, J. Park, C. Lee, and J.-J. Kim, Improving Accuracy of Binary Neural Networks using Unbalanced Activation Distribution. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 7858-7867 (2021).

[131] S. Merity, C. Xiong, J. Bradbury, and R. Socher, Pointer Sentinel Mixture Models. In *International Conference on Learning Representations (ICLR)* (2017).

[132] A. Chowdhery, et al. *PaLM: Scaling Language Modeling with Pathways*. (2022).

[133] S. Smith, M. Patwary, B. Norick, P. LeGresley, S. Rajbhandari, J. Casper, Z. Liu, S. Prabhumoye, G. Zerveas, V. Korthikanti, E. Zhang, R. Child, R. Y. Aminabadi, J. Bernauer, X. Song, M. Shoeybi, Y. He, M. Houston, S. Tiwary, and B. Catanzaro, *Using DeepSpeed and Megatron to Train Megatron-Turing NLG 530B, A Large-Scale Generative Language Model*. (2022).

[134] A. Ahmadian, S. Dash, H. Chen, B. Venkitesh, S. Gou, P. Blunsom, A. Üstün, and S. Hooker, Intriguing Properties of Quantization at Scale. *arXiv:2305.19268* (2023).

[135] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, A. Steiner, D. Keysers, J. Uszkoreit, M. Lucic, and A. Dosovitskiy, MLP-Mixer: An all-MLP Architecture for Vision. *arXiv preprint arXiv:2105.01601* (2021).

[136] H. Liu, Z. Dai, D. R. So, and Q. V. Le, *Pay Attention to MLPs*. (2021).

[137] S. Shankar and A. Reuther, Trends in Energy Estimates for Computing in AI/Machine Learning Accelerators, Supercomputers, and Compute-Intensive Applications. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)* (2022).

[138] S. Datta, W. Chakraborty, and M. Radosavljevic, Toward attojoule switching energy in logic transistors. *Science* **378**, 733–740 (2022).

[139] N. C. Menicucci, P. van Loock, M. Gu, C. Weedbrook, T. C. Ralph, and M. A. Nielsen, Universal Quantum Computation with Continuous-Variable Cluster States. *Phys. Rev. Lett.* **97**11, , 110501 (2006).

[140] S.-H. Tan, B. I. Erkmen, V. Giovannetti, S. Guha, S. Lloyd, L. Maccone, S. Pirandola, and J. H. Shapiro, Quantum Illumination with Gaussian States. *Phys. Rev. Lett.* **101**25, , 253601 (2008).

[141] C. Weedbrook, S. Pirandola, R. García-Patrón, N. J. Cerf, T. C. Ralph, J. H. Shapiro, and S. Lloyd, Gaussian quantum information. *Rev. Mod. Phys.* **84**2, , 621–669 (2012).

[142] J. E. Bourassa, R. N. Alexander, M. Vasmer, A. Patil, I. Tzitrin, T. Matsuura, D. Su, B. Q. Baragiola, S. Guha, G. Dauphinais, K. K. Sabapathy, N. C. Menicucci, and I. Dhand, Blueprint for a Scalable Photonic Fault-Tolerant Quantum Computer. *Quantum* **5**, 392 (2021).

[143] S. Rahimi-Keshari, A. P. Lund, and T. C. Ralph, What Can Quantum Optics Say about Computational Complexity Theory? *Phys. Rev. Lett.* **114**6, , 060501 (2015).

[144] C. S. Hamilton, R. Kruse, L. Sansoni, S. Barkhofen, C. Silberhorn, and I. Jex, Gaussian Boson Sampling. *Phys. Rev. Lett.* **119**17, , 170501 (2017).

[145] D. Hangleiter and J. Eisert, Computational advantage of quantum random sampling. *Rev. Mod. Phys.* **95**3, , 035001 (2023).

[146] H.-S. Zhong, et al. Quantum computational advantage using photons. *Science* **370**, 1460–1463 (2020).

[147] H.-S. Zhong, et al. Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light. *Phys. Rev. Lett.* **127**18, , 180502 (2021).

[148] Y.-H. Deng, et al. Gaussian Boson Sampling with Pseudo-Photon-Number-Resolving Detectors and Quantum Computational Advantage. *Phys. Rev. Lett.* **131**15, , 150601 (2023).

[149] L. S. Madsen, F. Laudenbach, M. F. Askarani, F. Rortais, T. Vincent, J. F. F. Bulmer, F. M. Miatto, L. Neuhaus, L. G. Helt, M. J. Collins, A. E. Lita, T. Gerrits, S. W. Nam, V. D. Vaidya, M. Menotti, I. Dhand, Z. Vernon, N. Quesada, and J. Lavoie, Quantum computational advantage with a programmable photonic processor. *Nature* **606**, 75–81 (2022).

[150] J. M. Lukens and P. Lougovski, Frequency-encoded photonic qubits for scalable quantum information processing. *Optica* **4**, 8–16 (2017).

[151] C. Joshi, A. Farsi, S. Clemmen, S. Ramelow, and A. L. Gaeta, Frequency multiplexing for quasi-deterministic heralded single-photon sources. *Nature Communications* **9** (2018).

[152] C. Joshi, A. Farsi, and A. Gaeta, Frequency-Domain Boson Sampling. In *Conference on Lasers and Electro-Optics*, FTu1F.1, San Jose, CA, United States (2017).

[153] T. Hiemstra, T. Parker, P. Humphreys, J. Tiedau, M. Beck, M. Karpiński, B. Smith, A. Eckstein, W. Kolthammer, and I. Walmsley, Pure Single Photons From Scalable Frequency Multiplexing. *Phys. Rev. Appl.* **14**1, , 014052 (2020).

[154] C. Reimer, M. Kues, P. Roztocki, B. Wetzel, F. Grazioso, B. E. Little, S. T. Chu, T. Johnston, Y. Bromberg, L. Caspani, D. J. Moss, and R. Morandotti, Generation of multiphoton entangled quantum states by means of integrated frequency combs. *Science* **351**, 1176–1180 (2016).

[155] M. Cabrejo-Ponce, A. L. M. Muniz, M. Huber, and F. Steinlechner, High-Dimensional Entanglement for Quantum Communication in the Frequency Domain. *Laser & Photonics Reviews* p. 2201010 (2023).

[156] J. R. Basani, M. Heuck, D. R. Englund, and S. Krastanov, All-Photonic Artificial Neural Network Processor Via Non-linear Optics. (2022).

[157] N. Liu, Y. Liu, J. Li, L. Yang, and X. Li, Generation of multi-mode squeezed vacuum using pulse pumped fiber optical parametric amplifiers. *Optics Express* **24**, 2125–2133 (2016).

[158] A. Omi, A. Hosaka, M. Tomita, Y. Yamagishi, K. Wakui, S. Niimura, K. Takahashi, M. Takeoka, and F. Kannari, Independently programmable frequency-multiplexed phase-sensitive optical parametric amplification in the optical telecommunication band. *Opt. Express* **29**, 21683–21697 (2021).

[159] Y. Yamagishi, A. Hosaka, K. Tanji, S. Kurimura, and F. Kannari, Arbitrary Mixing of Frequency-Range Multimode Quantum States Using Nonlinear Waveguide Crystals Based on Dispersion Engineering. In *OSA Nonlinear Optics 2021*, NM2B.4, Washington, DC, United States (2021).

[160] J. Roslund, R. M. de Araújo, S. Jiang, C. Fabre, and N. Treps, Wavelength-multiplexed quantum networks with ultrafast frequency combs. *Nature Photonics* **8**, 109–112 (2014).

[161] Y. Cai, J. Roslund, G. Ferrini, F. Arzani, X. Xu, C. Fabre, and N. Treps, Multimode entanglement in reconfigurable graph states using optical frequency combs. *Nature Communications* **8**, 15645 (2017).

[162] V. Roman-Rodriguez, D. Fainsin, G. L. Zanin, N. Treps, E. Diamanti, and V. Parigi, Multimode Squeezed State for Reconfigurable Quantum Networks at Telecommunication Wavelengths. (2023).

[163] Z. Yang, M. Jahanbozorgi, D. Jeong, S. Sun, O. Pfister, H. Lee, and X. Yi, A squeezed quantum microcomb on a chip. *Nature Communications* **12**, 4781 (2021).

[164] M. Jahanbozorgi, Z. Yang, S. Sun, H. Chen, R. Liu, B. Wang, and X. Yi, Generation of squeezed quantum microcombs with silicon nitride integrated photonic circuits. *Optica* **10**, 1100–1101 (2023).

[165] M. Pysher, Y. Miwa, R. Shahrokhshahi, R. Bloomer, and O. Pfister, Parallel Generation of Quadripartite Cluster Entanglement in the Optical Frequency Comb. *Phys. Rev. Lett.* **107**3, , 030505 (2011).

[166] M. Chen, N. C. Menicucci, and O. Pfister, Experimental Realization of Multipartite Entanglement of 60 Modes of a Quantum Optical Frequency Comb. *Phys. Rev. Lett.* **112**12, , 120505 (2014).

[167] Z. Xie, T. Zhong, S. Shrestha, X. Xu, J. Liang, Y.-X. Gong, J. C. Bienfang, A. Restelli, J. H. Shapiro, F. N. C. Wong, and C. W. Wong, Harnessing high-dimensional hyperentanglement through a biphoton frequency comb. *Nature Photonics* **9**, 536–542 (2015).

[168] K.-C. Chang, X. Cheng, M. C. Sarihan, A. K. Vinod, Y. S. Lee, T. Zhong, Y.-X. Gong, Z. Xie, J. H. Shapiro, F. N. C. Wong, and C. W. Wong, 648 Hilbert-space dimensionality in a biphoton frequency comb: entanglement of formation and Schmidt mode decomposition. *npj Quantum Information* **7**, 48 (2021).

[169] A. Eldan, O. Gilon, A. Lagimi, E. Forman, and A. Pe'er, Multiplexed Processing of Quantum Information Across an Ultra-wide Optical Bandwidth. (2023).

[170] H.-H. Lu, J. M. Lukens, N. A. Peters, O. D. Odele, D. E. Leaird, A. M. Weiner, and P. Lougovski, Electro-Optic Frequency Beam Splitters and Tritters for High-Fidelity Photonic Quantum Information Processing. *Phys. Rev. Lett.* **120**3, , 030502 (2018).

[171] H.-H. Lu, E. M. Simmerman, P. Lougovski, A. M. Weiner, and J. M. Lukens, Fully Arbitrary Control of Frequency-Bin Qubits. *Phys. Rev. Lett.* **125**12, , 120503 (2020).

[172] U. A. Javid, R. Lopez-Rios, J. Ling, A. Graf, J. Staffa, and Q. Lin, Chip-scale simulations in a quantum-correlated synthetic space. *Nature Photonics* **17**, 883–890 (2023).

[173] L. Yuan, A. Dutt, and S. Fan, Synthetic frequency dimensions in dynamically modulated ring resonators. *APL Photonics* **6**, 071102 (2021).

[174] X. Zhu, C.-H. Chang, C. González-Arciniegas, A. Pe'er, J. Higgins, and O. Pfister, Hypercubic cluster states in the phase-modulated quantum optical frequency comb. *Optica* **8**, 281–290 (2021).

[175] H.-H. Lu, M. Liscidini, A. L. Gaeta, A. M. Weiner, and J. M. Lukens, Frequency-bin photonic quantum information. *Optica* **10**, 1655–1671 (2023).

[176] C. Joshi, A. Farsi, A. Dutt, B. Y. Kim, X. Ji, Y. Zhao, A. M. Bishop, M. Lipson, and A. L. Gaeta, Frequency-Domain Quantum Interference with Correlated Photons from an Integrated Microresonator. *Phys. Rev. Lett.* **124**14, , 143601 (2020).

[177] C. M. Natarajan, M. G. Tanner, and R. H. Hadfield, Superconducting nanowire single-photon detectors: physics and applications. *Superconductor Science and Technology* **25**, 063001 (2012).

[178] I. Esmaeil Zadeh, J. Chang, J. W. N. Los, S. Gyger, A. W. Elshaari, S. Steinhauer, S. N. Dorenbos, and V. Zwiller, Superconducting nanowire single-photon detectors: A perspective on evolution, state-of-the-art, future developments, and applications. *Applied Physics Letters* **118**, 190502 (2021).

[179] H. Defienne, B. Ndagano, A. Lyons, and D. Faccio, Polarization entanglement-enabled quantum holography. *Nature Physics* **17**, 591–597 (2021).

[180] E. Bolduc, D. Faccio, and J. Leach, Acquisition of multiple photon pairs with an EMCCD camera. *Journal of Optics* **19**, 054006 (2017).

[181] H. Defienne, J. Zhao, E. Charbon, and D. Faccio, Full-field quantum imaging with a single-photon avalanche diode camera. *Phys. Rev. A* **103**4, , 042608 (2021).

[182] A. Kumar and A. M. Marino, Spatial squeezing in bright twin beams generated with four-wave mixing: Constraints on characterization with an electron-multiplying charge-coupled-device camera. *Phys. Rev. A* **100**6, , 063828 (2019).

[183] F. Li, T. Li, M. O. Scully, and G. S. Agarwal, Quantum Advantage with Seeded Squeezed Light for Absorption Measurement. *Phys. Rev. Appl.* **15**4, , 044030 (2021).

[184] P. Svihra, Y. Zhang, P. Hockett, S. Ferrante, B. Sussman, D. England, and A. Nomerotski, Multivariate discrimination in quantum target detection. *Applied Physics Letters* **117**, 044001 (2020).

[185] K. Dhimitri, S. M. Fullerton, B. Coyle, K. E. Bennett, T. Miura, T. Higuchi, and T. Maruno, Scientific CMOS (sCMOS) camera capabilities with a focus on quantum applications. In *Photonics for Quantum 2022*, PC122430L ed. D. F. Figer,, Rochester, NY, United States Vol. PC12243 (2022).

[186] J. Ma, S. Masoodian, D. A. Starkey, and E. R. Fossum, Photon-number-resolving megapixel image sensor at room temperature without avalanche gain. *Optica* **4**, 1474–1481 (2017).

[187] E. Lantz, J.-L. Blanchet, L. Furfaro, and F. Devaux, Multi-imaging and Bayesian estimation for photon counting with EMCCDs. *Monthly Notices of the Royal Astronomical Society* **386**, 2262–2270 (2008).

[188] S. Olivares, Quantum optics in the phase space: a tutorial on Gaussian states. *The European Physical Journal Special Topics* **203**, 3–24 (2012).

[189] A. Ferraro, S. Olivares, and M. G. A. Paris, Gaussian states in continuous variable quantum information. (2005).

[190] W. Wasilewski, A. I. Lvovsky, K. Banaszek, and C. Radzewicz, Pulsed squeezed light: Simultaneous squeezing of multiple modes. *Phys. Rev. A* **73**6, , 063819 (2006).

[191] T. Opatrný, N. Korolkova, and G. Leuchs, Mode structure and photon number correlations in squeezed quantum pulses. *Phys. Rev. A* **66**5, , 053813 (2002).

[192] A. Hosaka, T. Kawamori, and F. Kannari, Multimode quantum theory of nonlinear propagation in optical fibers. *Phys. Rev. A* **94**5, , 053833 (2016).

[193] G. Adesso, S. Ragy, and A. R. Lee, Continuous variable quantum information: Gaussian states and beyond. *Open Systems & Information Dynamics* **21**, 1440001 (2014).

[194] E. R. Caianiello, On quantum field theory – I: explicit solution of Dyson's equation in electrodynamics without use of Feynman graphs. *Nuovo Cim.* **10**, 1634–1652 (1953).

[195] K. E. Cahill and R. J. Glauber, Ordered Expansions in Boson Amplitude Operators. *Phys. Rev.* **177**5, , 1857–1881 (1969).

[196] J. Katriel, Combinatorial aspects of boson algebra. *Lett. Nuovo Cimento* **10**, 565–567 (1974).

[197] S. Aaronson and A. Arkhipov, The Computational Complexity of Linear Optics. In *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11, New York, NY, USA, 333–342 (2011).

[198] H.-Y. Fan, Operator ordering in quantum optics theory and the development of Dirac's symbolic method. *Journal of Optics B: Quantum and Semiclassical Optics* **5**, R147–R163 (2003).

[199] Nuvu Cameras, *HNü 512 – 512 x 512 EMCCD* (`https://www.nuvucameras.com/products/hnu-512/`). Accessed: 2024-01-07. (year?).

[200] M. J. Fitch, B. C. Jacobs, T. B. Pittman, and J. D. Franson, Photon-number resolution using time-multiplexed single-photon detectors. *Phys. Rev. A* **68**4, , 043814 (2003).

[201] H. Paul, P. Törmä, T. Kiss, and I. Jex, Photon Chopping: New Way to Measure the Quantum State of Light. *Phys. Rev. Lett.* **76**14, , 2464–2467 (1996).

[202] J. Sperling, W. Vogel, and G. S. Agarwal, True photocounting statistics of multiple on-off detectors. *Phys. Rev. A* **85**2, , 023820 (2012).

[203] H. Qi, D. J. Brod, N. Quesada, and R. García-Patrón, Regimes of classical simulability for noisy Gaussian boson sampling. *Phys. Rev. Lett.* **124**10, , 100502 (2020).

[204] P. Kumar, Quantum frequency conversion. *Opt. Lett.* **15**, 1476–1478 (1990).

[205] X. Wang, X. Jiao, B. Wang, Y. Liu, X.-P. Xie, M.-Y. Zheng, Q. Zhang, and J.-W. Pan, Quantum frequency conversion and single-photon detection with lithium niobate nanophotonic chips. *npj Quantum Information* **9** (2023).

[206] C. E. Vollmer, C. Baune, A. Samblowski, T. Eberle, V. Händchen, J. Fiurášek, and R. Schnabel, Quantum Up-Conversion of Squeezed Vacuum States from 1550 to 532 nm. *Phys. Rev. Lett.* **112**7, , 073602 (2014).

[207] C. Baune, J. Gniesmer, A. Schönbeck, C. E. Vollmer, J. Fiurášek, and R. Schnabel, Strongly squeezed states at 532 nm based on frequency upconversion. *Opt. Express* **23**, 16035–16041 (2015).

[208] A. Samblowski, C. E. Vollmer, C. Baune, J. Fiurášek, and R. Schnabel, Weak-signal conversion from 1550 to 532 nm with 84% efficiency. *Opt. Lett.* **39**, 2979–2981 (2014).

[209] K. A. G. Bonsma-Fisher, P. J. Bustard, C. Parry, T. A. Wright, D. G. England,

B. J. Sussman, and P. J. Mosley, Ultratunable Quantum Frequency Conversion in Photonic Crystal Fiber. *Phys. Rev. Lett.* **129**20, , 203603 (2022).

[210] M. Allgaier, V. Ansari, L. Sansoni, C. Eigner, V. Quiring, R. Ricken, G. Harder, B. Brecht, and C. Silberhorn, Highly efficient frequency conversion with bandwidth compression of quantum light. *Nature Communications* **8** (2017).

[211] H. Suchowski, B. D. Bruner, Y. Israel, A. Ganany-Padowicz, A. Arie, and Y. Silberberg, Broadband photon pair generation at $3\omega/2$. *Applied Physics B* **122**, 1–5 (2016).

[212] H. Suchowski, V. Prabhudesai, D. Oron, A. Arie, and Y. Silberberg, Robust adiabatic sum frequency conversion. *Opt. Express* **17**, 12731–12740 (2009).

[213] J. Moses, H. Suchowski, and F. X. Kärtner, Fully efficient adiabatic frequency conversion of broadband Ti:sapphire oscillator pulses. *Opt. Lett.* **37**, 1589–1591 (2012).

[214] C. Beenakker and C. Schönenberger, Quantum shot noise. *Physics Today* **56**, 37–42 (2003).

[215] G. S. Agarwal, (2012) *Quantum Optics*. (Cambridge University Press).

[216] S. Machida, Y. Yamamoto, and Y. Itaya, Observation of amplitude squeezing in a constant-current–driven semiconductor laser. *Physical Review Letters* **58**, 1000 (1987).

[217] R. H. Hadfield, Single-photon detectors for optical quantum information applications. *Nature Photonics* **3**, 696–705 (2009).

[218] A. Alaghi and J. P. Hayes, Survey of stochastic computing. *ACM Transactions on Embedded Computing Systems (TECS)* **12**, 1–19 (2013).

[219] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, A learning algorithm for Boltzmann machines. *Cognitive Science* **9**, 147–169 (1985).

[220] R. M. Neal, Learning stochastic feedforward networks. *Department of Computer Science, University of Toronto* **64**, 1577 (1990).

[221] R. M. Neal, Connectionist learning of belief networks. *Artificial Intelligence* **56**, 71–113 (1992).

[222] Y. Bengio, N. Léonard, and A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv:1308.3432* (2013).

[223] C. Tang and R. R. Salakhutdinov, Learning stochastic feedforward neural networks. *Advances in Neural Information Processing Systems* **26** (2013).

[224] T. Raiko, M. Berglund, G. Alain, and L. Dinh, Techniques for learning binary stochastic feedforward neural networks. *arXiv:1406.2989* (2014).

[225] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, Binarized neural networks. *Advances in Neural Information Processing Systems* **29** (2016).

[226] Y. Ji, F. Ran, C. Ma, and D. J. Lilja, A hardware implementation of a radial basis function neural network using stochastic logic. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 880–883 (2015).

[227] V. T. Lee, A. Alaghi, J. P. Hayes, V. Sathe, and L. Ceze, Energy-efficient hybrid stochastic-binary neural networks for near-sensor computing. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 13–18 (2017).

[228] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, A survey of stochastic computing neural networks for machine learning applications. *IEEE Transactions on Neural Networks and Learning Systems* **32**, 2809–2824 (2020).

[229] D. Vodenicarevic, N. Locatelli, A. Mizrahi, J. S. Friedman, A. F. Vincent, M. Romera, A. Fukushima, K. Yakushiji, H. Kubota, S. Yuasa et al. Low-energy truly random number generation with superparamagnetic tunnel junctions for unconventional computing. *Physical Review Applied* **8**, 054045 (2017).

[230] O. Hassan, R. Faria, K. Y. Camsari, J. Z. Sun, and S. Datta, Low-barrier magnet design for efficient hardware binary stochastic neurons. *IEEE Magnetics Letters* **10**, 1–5 (2019).

[231] S. Chowdhury, A. Grimaldi, N. A. Aadit, S. Niazi, M. Mohseni, S. Kanai, H. Ohno, S. Fukami, L. Theogarajan, G. Finocchio et al. A full-stack view of probabilistic computing with p-bits: devices, architectures and algorithms. *IEEE Journal on Exploratory Solid-State Computational Devices and Circuits* (2023).

[232] T. Hylton, T. M. Conte, and M. D. Hill, A vision to compute like nature: Thermodynamically. *Communications of the ACM* **64**, 35–38 (2021).

[233] P. J. Coles, C. Szczepanski, D. Melanson, K. Donatella, A. J. Martinez, and F. Sbahi, Thermodynamic AI and the fluctuation frontier. *arXiv:2302.06584* (2023).

[234] C. Wu, X. Yang, Y. Chen, and M. Li, Photonic Bayesian neural network using programmed optical noises. *IEEE Journal of Selected Topics in Quantum Electronics* **29**, 1–6 (2022).

[235] B. Ma, J. Zhang, X. Li, and W. Zou, Stochastic photonic spiking neuron for Bayesian inference with unsupervised learning. *Optics Letters* **48**, 1411–1414 (2023).

[236] S. Gu, S. Levine, I. Sutskever, and A. Mnih, MuProp: Unbiased backpropagation for stochastic neural networks. *arXiv:1511.05176* (2015).

[237] Y. Liu, S. Liu, Y. Wang, F. Lombardi, and J. Han, A stochastic computational multi-layer perceptron with backward propagation. *IEEE Transactions on Computers* **67**, 1273–1286 (2018).

[238] C. Gerry and P. Knight, (2004) *Introductory Quantum Optics*. (Cambridge University Press, Cambridge).

[239] L. Bottou, Stochastic gradient descent tricks. *Neural Networks: Tricks of the Trade: Second Edition* pp. 421–436 (2012).

[240] I. Loshchilov and F. Hutter, Decoupled weight decay regularization. *arXiv:1711.05101* (2017).

[241] C.-Y. Lee, P. W. Gallagher, and Z. Tu, Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, 464–472 (2016).

[242] S. K. Esser, P. A. Merolla, J. V. Arthur, A. S. Cassidy, R. Appuswamy, A. Andreopoulos, D. J. Berg, J. L. McKinstry, T. Melano, D. R. Barch, C. di Nolfo, P. Datta, A. Amir, B. Taba, M. D. Flickner, and D. S. Modha, Convolutional networks for fast, energy-efficient neuromorphic computing. *Proceedings of the National Academy of Sciences* **113**, 11441–11446 (2016).

[243] H. Qin, R. Gong, X. Liu, X. Bai, J. Song, and N. Sebe, Binary neural networks: A survey. *Pattern Recognition* **105**, 107281 (2020).

[244] A. Krizhevsky, Learning multiple layers of features from tiny images. *Technical Report* pp. 32–33 (2009).

[245] T. Clanuwat and et al., Deep Learning for Classical Japanese Literature. *arXiv preprint arXiv:1812.01718* (2018).

[246] H. Xiao and et al., Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[247] K. Dhimitri, S. M. Fullerton, B. Coyle, K. E. Bennett, T. Miura, T. Higuchi, and T. Maruno, Scientific CMOS (sCMOS) camera capabilities with a focus on quantum applications. In *Photonics for Quantum 2022*, PC122430L (2022).

[248] C. Wu, X. Yang, H. Yu, R. Peng, I. Takeuchi, Y. Chen, and M. Li, Harnessing Optoelectronic Noises in a Hybrid Photonic Generative Adversarial Network (GAN). *Preprint at Research Square:10.21203* (2021).

[249] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, XNOR-Net: Imagenet classification using binary convolutional neural networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV*, 525–542 (2016).

[250] A. Bulat, J. Kossaifi, G. Tzimiropoulos, and M. Pantic, Matrix and tensor decompositions for training binary neural networks. *arXiv:1904.07852* (2019).

[251] J. Torrejon, M. Riou, F. A. Araujo, S. Tsunegi, G. Khalsa, D. Querlioz, P. Bortolotti, V. Cros, K. Yakushiji, A. Fukushima et al. Neuromorphic computing with nanoscale spintronic oscillators. *Nature* **547**, 428–431 (2017).

[252] J. Grollier, D. Querlioz, K. Camsari, K. Everschor-Sitte, S. Fukami, and M. D. Stiles, Neuromorphic spintronics. *Nature Electronics* **3**, 360–370 (2020).

[253] F. Cai, S. Kumar, T. Van Vaerenbergh, X. Sheng, R. Liu, C. Li, Z. Liu, M. Foltin, S. Yu, Q. Xia et al. Power-efficient combinatorial optimization

using intrinsic noise in memristor Hopfield neural networks. *Nature Electronics* **3**, 409–418 (2020).

[254] K.-E. Harabi, T. Hirtzlin, C. Turck, E. Vianello, R. Laurent, J. Droulez, P. Bessière, J.-M. Portal, M. Bocquet, and D. Querlioz, A memristor-based Bayesian machine. *Nature Electronics* **6**, 52–63 (2023).

[255] A. N. M. N. Islam, K. Yang, A. K. Shukla, P. Khanal, B. Zhou, W.-G. Wang, and A. Sengupta, Hardware in Loop Learning with Spin Stochastic Neurons. *arXiv:2305.03235* (2023).

[256] D. Marković and J. Grollier, Quantum neuromorphic computing. *Applied Physics Letters* **117** (2020).

[257] M. Cerezo, G. Verdon, H.-Y. Huang, L. Cincio, and P. J. Coles, Challenges and opportunities in quantum machine learning. *Nature Computational Science* **2**, 567–576 (2022).

[258] C. Roques-Carmes, Y. Salamin, J. Sloan, S. Choi, G. Velez, E. Koskas, N. Rivera, S. E. Kooi, J. D. Joannopoulos, and M. Soljačić, Biasing the quantum vacuum to control macroscopic probability distributions. *Science* **381**, 205–209 (2023).

[259] M. Prezioso, F. Merrikh-Bayat, B. D. Hoskins, G. C. Adam, K. K. Likharev, and D. B. Strukov, Training and operation of an integrated neuromorphic network based on metal-oxide memristors. *Nature* **521**, 61–64 (2015).

[260] M. Romera, P. Talatchian, S. Tsunegi, F. Abreu Araujo, V. Cros, P. Bortolotti, J. Trastoy, K. Yakushiji, A. Fukushima, H. Kubota et al. Vowel recognition with four coupled spin-torque nano-oscillators. *Nature* **563**, 230–234 (2018).

[261] K. Mitarai, M. Negoro, M. Kitagawa, and K. Fujii, Quantum circuit learning. *Physical Review A* **98**, 032309 (2018).

[262] T. W. Hughes, I. A. Williamson, M. Minkov, and S. Fan, Wave physics as an analog recurrent neural network. *Science Advances* **5**, eaay6946 (2019).

[263] T. Chen, J. van Gelder, B. van de Ven, S. V. Amitonov, B. De Wilde, H.-C. Ruiz Euler, H. Broersma, P. A. Bobbert, F. A. Zwanenburg, and W. G. van der Wiel, Classification with a disordered dopant-atom network in silicon. *Nature* **577**, 341–345 (2020).

[264] B. Cramer, S. Billaudelle, S. Kanya, A. Leibfried, A. Grübl, V. Karasenko, C. Pehle, K. Schreiber, Y. Stradmann, J. Weis et al. Surrogate gradients for analog neuromorphic computing. *Proceedings of the National Academy of Sciences* **119**, e2109194119 (2022).

[265] A. Ross, N. Leroux, A. De Riz, D. Marković, D. Sanz-Hernández, J. Trastoy, P. Bortolotti, D. Querlioz, L. Martins, L. Benetti et al. Multilayer spintronic neural networks with radiofrequency connections. *Nature Nanotechnology* pp. 1–8 (2023).

[266] A. N. Tait, Quantifying Power in Silicon Photonic Neural Networks. *Phys. Rev. Appl.* **17**5, , 054029 (2022).

[267] M. Rani, S. B. Dhok, and R. B. Deshmukh, A systematic review of compressive sensing: Concepts, implementations and applications. *IEEE access* **6**, 4875–4894 (2018).

[268] G. M. Gibson, S. D. Johnson, and M. J. Padgett, Single-pixel imaging 12 years on: a review. *Optics express* **28**, 28190–28208 (2020).

[269] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, Neural sensors: Learning pixel exposures for HDR imaging and video compressive sensing with programmable sensors. *IEEE transactions on pattern analysis and machine intelligence* **42**, 1642–1653 (2020).

[270] P. Pad, S. Narduzzi, C. Kundig, E. Turetken, S. A. Bigdeli, and L. A. Dunbar, Efficient neural vision systems based on convolutional image acquisition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12285–12294 (2020).

[271] H. Zheng, Q. Liu, Y. Zhou, I. I. Kravchenko, Y. Huo, and J. Valentine, Meta-optic accelerators for object classifiers. *Science Advances* **8**, eabo6410 (2022).

[272] I. Esmaeil Zadeh, J. W. Los, R. B. Gourgues, J. Chang, A. W. Elshaari, J. R. Zichi, Y. J. Van Staaden, J. P. Swens, N. Kalhor, A. Guardiani et al. Efficient single-photon detection with 7.7 ps time resolution for photon-correlation measurements. *Acs Photonics* **7**, 1780–1787 (2020).

[273] Y. Wang, (2022) Ph.D. thesis (Macquarie University).

[274] G. van den Engh and C. Farmer, Photo-bleaching and photon saturation

in flow cytometry. *Cytometry: The Journal of the International Society for Analytical Cytology* **13**, 669–677 (1992).

[275] R. Nehra, R. Sekine, L. Ledezma, Q. Guo, R. M. Gray, A. Roy, and A. Marandi, Few-cycle vacuum squeezing in nanophotonics. *Science* **377**, 1333–1337 (2022).

[276] L. Ledezma, R. Sekine, Q. Guo, R. Nehra, S. Jahani, and A. Marandi, Intense optical parametric amplification in dispersion-engineered nanophotonic lithium niobate waveguides. *Optica* **9**, 303–308 (2022).

[277] I. Hurvitz, A. Karnieli, and A. Arie, Frequency-domain engineering of bright squeezed vacuum for continuous-variable quantum information. *Optics Express* **31**, 20387 (2023).

[278] C. Drago and A. M. Brańczyk, Tunable frequency-bin multimode squeezed vacuum states of light. *Phys. Rev. A* **106**4, , 043714 (2022).

[279] C. L. Morrison, F. Graffitti, P. Barrow, A. Pickston, J. Ho, and A. Fedrizzi, Frequency-bin entanglement from domain-engineered down-conversion. *APL Photonics* **7**, 066102 (2022).

[280] P. Folge, M. Santandrea, M. Stefszky, B. Brecht, and C. Silberhorn, Multimode Squeezed States in Frequency Conversion Based Time-Frequency Networks. In *Frontiers in Optics + Laser Science 2022 (FIO, LS)*, FM5B.5, Rochester, NY, United States (2022).

[281] A. Bulat and G. Tzimiropoulos, XNOR-Net++: Improved binary neural networks. *arXiv:1909.13863* (2019).

[282] N. Mohseni, P. L. McMahon, and T. Byrnes, Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics* **4**, 363–379 (2022).

[283] C. Bruschini, S. Burri, E. Bernasconi, T. Milanese, A. C. Ulku, H. Homulle, and E. Charbon, LinoSPAD2: A 512x1 linear SPAD camera with system-level 135-ps SPTR and a reconfigurable computational engine for time-resolved single-photon imaging. In *Quantum Sensing and Nano Electronics and Photonics XIX* Vol. 12430, 126–135 (2023).

[284] J. M. Shainline, S. M. Buckley, R. P. Mirin, and S. W. Nam, Superconduct-

ing optoelectronic circuits for neuromorphic computing. *Physical Review Applied* **7**, 034013 (2017).

[285] A. H. Kiilerich and K. Mølmer, Input-output theory with quantum pulses. *Physical Review Letters* **123**, 123604 (2019).

[286] Q. Li, K. Orcutt, R. L. Cook, J. Sabines-Chesterking, A. L. Tong, G. S. Schlau-Cohen, X. Zhang, G. R. Fleming, and K. B. Whaley, Single-photon absorption and emission from a natural photosynthetic complex. *Nature* pp. 1–5 (2023).

[287] L. Li, L. D. Santis, I. B. Harris, K. C. Chen, Y. Gao, I. Christen, H. Choi, M. Trusheim, Y. Song, C. Errando-Herranz et al. Heterogeneous integration of spin–photon interfaces with a CMOS platform. *Nature* pp. 1–7 (2024).

[288] V. Saggio, C. Errando-Herranz, S. Gyger, C. Panuski, M. Prabhu, L. De Santis, I. Christen, D. Ornelas-Huerta, H. Raniwala, C. Gerlach et al. Cavity-enhanced single artificial atoms in silicon. *Nature Communications* **15**, 5296 (2024).

[289] T. Zhou, L. Fang, T. Yan, J. Wu, Y. Li, J. Fan, H. Wu, X. Lin, and Q. Dai, In situ optical backpropagation training of diffractive optical neural networks. *Photonics Research* **8**, 940–953 (2020).

[290] X. Guo, T. D. Barrett, Z. M. Wang, and A. Lvovsky, Backpropagation through nonlinear units for the all-optical training of neural networks. *Photonics Research* **9**, B71–B80 (2021).

[291] S. Pai, Z. Sun, T. W. Hughes, T. Park, B. Bartlett, I. A. Williamson, M. Minkov, M. Milanizadeh, N. Abebe, F. Morichetti et al. Experimentally realized in situ backpropagation for deep learning in photonic neural networks. *Science* **380**, 398–404 (2023).

[292] Y. Bengio, D.-H. Lee, J. Bornschein, T. Mesnard, and Z. Lin, Towards biologically plausible deep learning. *arXiv:1502.04156* (2015).

[293] T. P. Lillicrap, A. Santoro, L. Marris, C. J. Akerman, and G. Hinton, Backpropagation and the brain. *Nature Reviews Neuroscience* **21**, 335–346 (2020).

[294] G. Hinton, The concept of mortal computation. Keynote address pre-

sented at the Neural Information Processing Systems conference, New Orleans (2023).

[295] M. Stern and A. Murugan, Learning without neurons in physical systems. *Annual Review of Condensed Matter Physics* **14**, 417–441 (2023).

[296] K. Berggren, Q. Xia, K. K. Likharev, D. B. Strukov, H. Jiang, T. Mikolajick, D. Querlioz, M. Salinga, J. R. Erickson, S. Pi et al. Roadmap on emerging hardware and technology for machine learning. *Nanotechnology* **32**, 012002 (2020).

[297] G. Finocchio, S. Bandyopadhyay, P. Lin, G. Pan, J. J. Yang, R. Tomasello, C. Panagopoulos, M. Carpentieri, V. Puliafito, J. kerman et al. Roadmap for unconventional computing with nanotechnology. *arXiv:2301.06727* (2023).

[298] C. E. Leiserson, N. C. Thompson, J. S. Emer, B. C. Kuszmaul, B. W. Lampson, D. Sanchez, and T. B. Schardl, There's plenty of room at the Top: What will drive computer performance after Moore's law? *Science* **368**, eaam9744 (2020).

[299] M. Kellman, M. Lustig, and L. Waller, How to do physics-based learning. *arXiv:2005.13531* (2020).

[300] B. Abbott, et al. Observation of Gravitational Waves from a Binary Black Hole Merger. *Physical Review Letters* **116** (2016).

[301] S. Pirandola, B. R. Bardhan, T. Gehring, C. Weedbrook, and S. Lloyd, Advances in photonic quantum sensing. *Nature Photonics* **12**, 724–733 (2018).

[302] C. Cui, W. Horrocks, S. Hao, S. Guha, N. Peyghambarian, Q. Zhuang, and Z. Zhang, Quantum receiver enhanced by adaptive learning. *Light: Science & Applications* **11**, 344 (2022).

[303] J. Qin, Y.-H. Deng, H.-S. Zhong, L.-C. Peng, H. Su, Y.-H. Luo, J.-M. Xu, D. Wu, S.-Q. Gong, H.-L. Liu et al. Unconditional and robust quantum metrological advantage beyond N00N states. *Physical Review Letters* **130**, 070801 (2023).

[304] H. Shi, Z. Zhang, S. Pirandola, and Q. Zhuang, Entanglement-assisted absorption spectroscopy. *Physical Review Letters* **125**, 180502 (2020).

[305] X. Gao, E. R. Anschuetz, S.-T. Wang, J. I. Cirac, and M. D. Lukin, Enhancing generative models via quantum correlations. *Physical Review X* **12**, 021037 (2022).

[306] E. R. Anschuetz, H.-Y. Hu, J.-L. Huang, and X. Gao, Interpretable quantum advantage in neural sequence learning. *PRX Quantum* **4**, 020338 (2023).

[307] J. M. Arrazola, et al. Quantum circuits with many photons on a programmable nanophotonic chip. *Nature* **591**, 54–60 (2021).

[308] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley, (1984) *Boltzmann machines: Constraint satisfaction networks that learn*. (Carnegie-Mellon University, Department of Computer Science Pittsburgh, PA).

[309] R. J. Williams, Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* **8**, 229–256 (1992).

[310] L. Weaver and N. Tao, The optimal reward baseline for gradient-based reinforcement learning. *arXiv:1301.2315* (2013).

[311] I. R. Fiete and H. S. Seung, Gradient learning in spiking neural networks by dynamic perturbation of conductances. *Physical Review Letters* **97**, 048104 (2006).

[312] G. Hinton, *Neural netowrks for machine learning*. Coursera, Video Lectures (2012).

[313] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, Understanding straight-through estimator in training activation quantized neural nets. *arXiv:1903.05662* (2019).

[314] J. Chung, S. Ahn, and Y. Bengio, Hierarchical multiscale recurrent neural networks. *arXiv:1609.01704* (2016).

[315] D. P. Kingma and J. Ba, Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).

[316] P. Ramachandran, B. Zoph, and Q. V. Le, Searching for activation functions. *arXiv:1710.05941* (2017).