Department of Physics and Astronomy
Heidelberg University

Bachelor Thesis in Physics
submitted by

Simon Tebeck

born 31. August 2001
in Bitburg (Germany)

June 2025

Machine Learning Applications on an Analog Photonic Hardware Accelerator

This Bachelor Thesis has been carried out by Simon Tebeck at the
Kirchhoff-Institut für Physik (KIP) of the
Ruprecht-Karls Universität Heidelberg
under the supervision of
Prof. Dr. Wolfram Pernice

Abstract

Analog photonic hardware accelerators offer a promising pathway to meet the growing demand for highly parallel and energy-efficient computation in deep learning. In this study, a photonic tensor core was investigated and used to perform inference on the MNIST and CIFAR-10 datasets. New control algorithms were developed to enable fast and accurate weight setting and output scaling while compensating for analog non-idealities. Based on these algorithms, functions for matrix-vector multiplications (MVMs) and convolutions were implemented within a machine learning framework. Two convolutional neural networks (CNNs) were pre-trained and fine-tuned in a hardware-aware manner to meet the system's constraints. These models were then deployed on the photonic integrated system and tested on image classification tasks. A shallow CNN consistently reached over 99.0% accuracy on MNIST with all MVMs and convolutions computed optically, matching the digital baseline. For CIFAR-10, a deeper CNN achieved 85.0% accuracy on a 500-image subset, with all convolutions photonically processed and only the final linear layer computed digitally – scoring just 1.1 percentage points below the floating-point reference. Although externally limited in speed, these results demonstrate the practical feasibility of multi-layer neural networks on photonic tensor cores, and the developed tools provide a foundation for further experimental research.

Zusammenfassung

Analoge photonische Hardwarebeschleuniger bieten einen vielversprechenden Ansatz, um die wachsende Nachfrage nach energieeffizienten und parallelen Hochgeschwindigkeitsberechnungen für Deep Learning zu erfüllen. In dieser Arbeit wurde ein photonischer Tensor-Kern untersucht und verwendet, um Bilderkennungsaufgaben auf den MNIST- und CIFAR-10-Datensätzen durchzuführen. Dazu wurden Steuerungsalgorithmen entwickelt, die ein schnelles und präzises Setzen von Gewichten und Skalieren von Ausgangssignalen ermöglichen und dabei analoge Nichtidealitäten kompensieren. Darauf aufbauend wurden Funktionen für optische Matrix-Vektor-Multiplikationen (MVMs) und Faltungen (Convolutions) implementiert, abgestimmt auf Anwendungen im maschinellen Lernen. Zwei Convolutional Neural Networks (CNNs) wurden trainiert und anschließend hardwarenah feinjustiert, um unter den Systemgegebenheiten robuster zu funktionieren. Diese Modelle wurden dann auf dem integrierten photonischen Aufbau eingesetzt und getestet. Ein flaches CNN, bei dem alle MVMs und Faltungen optisch berechnet wurden, erzielte bei MNIST wiederholt Genauigkeiten von über 99,0% und entsprach damit der simulierten Referenz. Bei CIFAR-10 erreichte ein tieferes CNN auf einer Teilmenge von 500 Bildern 85,0% Genauigkeit, wobei alle Faltungen photonisch verarbeitet und nur die letzte lineare Schicht digital berechnet wurde. Dieses Ergebnis lag lediglich 1,1 Prozentpunkte unter der Gleitkommareferenz. Trotz externer Geschwindigkeitsbegrenzungen zeigen diese Resultate die praktische Durchführbarkeit mehrschichtiger neuronaler Netze auf photonischen Tensor-Kernen. Die entwickelten Methoden bieten eine Grundlage für weiterführende Forschung.

Contents

1. Introduction

Machine Learning with Light

Machine learning is a subfield of artificial intelligence that emerged alongside the development of early computers and has been explored in its modern sense since the 1950s [15]. Recent breakthroughs, enabled by significantly improved hardware, vast data availability, and foundational advances in model architectures and training algorithms, have generated widespread public interest over the last few years. Prominent examples include freely available large language models such as ChatGPT or Gemini. However, current machine learning applications require substantial computational resources and consume large amounts of energy. It is estimated that the computational demand for deep neural networks (DNNs) doubles every four to six months [29], posing challenges to the energy infrastructure and causing a notable carbon footprint [22][4]. This has fueled the search for alternative technologies that overcome the limitations of classical von Neumann architectures, which fundamentally rely on serialized operations and physically inefficient separation of memory and processing units.

Photonic analog computing presents a promising pathway to address both the growing demand for fast and massively parallelized calculations, as well as the need for more energy-efficient training and inference [34]. At its core, the concept leverages light to encode information in its physical properties, such as intensity, phase, or polarization, and to perform mathematical operations by exploiting mechanisms like interference or absorption. In telecommunication, optical signal transmission has been widely deployed for decades, achieving ultra-high bandwidth, high speed, and low-latency [34]. To harness these advantages for computations on photonic integrated circuits (PICs), several optical processor designs have been proposed, including Mach-Zehnder interferometer networks [25] and microring resonator banks [28]. In this study, we will explore the approach of photonic tensor cores: They implement a crossbar structure in which incoming light is distributed over multiple input channels to nodes that encode the weights, thereby, for instance, using phase-change materials [14] or electro-absorption modulators (EAMs) [10]. At these nodes, multiply-accumulate (MAC) operations are performed, which form the basis for larger matrix-vector multiplications (MVMs). In machine learning, MVMs constitute the dominant computational operation and are responsible for the major time and power consumption, which is why their acceleration with photonic hardware has great

potential for future DNN applications [29]. However, like other analog computing devices, photonic systems must contend with noise, limited precision and various hardware non-idealities. Moreover, due to their experimental state, size limitations of current PICs and speed constraints induced by their electronic support infrastructure pose further challenges [29].

Scope and Structure of this Work

In this Bachelor's thesis, an existing integrated system featuring a photonic tensor core was analyzed and utilized to perform basic machine learning tasks. To enhance the performance of the setup, algorithms were developed that improve the weight setting process while accounting for common non-idealities, such as non-linear scaling, channel inhomogenities, and crosstalk. Building upon the system's software interfaces, frontend functions were implemented that perform not only MVMs but also convolutions, a key element of artificial neural networks used in computer vision problems. To cope with the noise and other hardware characteristics, such as quantization, two selected machine learning models were initially trained conventionally and then fine-tuned in a hardware-aware manner. These models were then deployed on the optical setup and tested for their inference capabilities on the MNIST and CIFAR-10 benchmarks.
The utilized PIC is still a prototype, therefore its crossbar array is relatively small, and external components currently limit its operational speed. As a consequence, computation times are not yet competitive with digital processing. Nevertheless, this study serves as a proof of concept and demonstrates that the introduced methods enable photonic machine learning, such as multi-layer convolutional neural networks.

The report is structured as follows: First, Chapter 2 lays the theoretical foundation of this work, explaining the principles of photonic tensor cores and giving a brief introduction to neural networks. Afterwards, Chapter 3 provides a more detailed description of the experimental setup at hand, while also discussing the newly developed control algorithms. Furthermore, the design of the machine learning models and the methodology for hardware-aware training and subsequent deployment on the photonic chip are outlined. The results are presented in Chapter 4, including evaluations of the developed functions, the fine-tuning process, and the photonic of photonic machine learning inference. Finally, Chapter 5 draws a conclusion and offers an outlook for future research. Supplementary figures can be found in the Appendix A.

2. Theoretical Background

2.1. Fundamentals of Analog Optical Processing

2.1.1. Photonic Tensor Core

The architecture of the *photonic tensor core* was first introduced by Feldmann et al. in 2021, proposing a specific crossbar unit on a photonic integrated circuit (PIC) that can perform high-speed analog optical matrix-vector multiplications with laser light from an external source [14]. The presented design combines fundamental building blocks of optical processing to couple, guide, and modulate light on-chip and arranges a weight structure based on phase-change material, capable of calculating trillions of multiply-accumulate operations (MACs) per second. In this work, experiments will be conducted on a similar photonic tensor core; its basic principles will be explained in the following paragraphs.

Concept

The fundamental operation carried out inside a photonic processing unit, but also in application-specific integrated circuits (ASICs) in general, is a multiply-accumulate operation, which includes the multiplication of two numbers and their addition to an accumulator. Multiplication with a factor is equivalent to modulating a characteristic of the electromagnetic wave - typically its phase or amplitude - while the accumulation happens by incoherent intensity summation of multiple signals [20]. Figure 2.1 shows a typical photonic crossbar array of size $m \times n$ in which each intersection carries out such a MAC operation. The accumulation happens over the column of the array, thus realizing an m-dimensional dot product

$$Y_j = \sum_{i=1}^{m} X_i a_{i,j} \tag{2.1}$$

per column. With n columns in total, the crossbar can calculate a matrix-vector multiplication (MVM) of size $(1 \times m) \cdot (m \times n)$ per time step. By splitting an input matrix into vectors and spreading them over the time domain, matrix-matrix multiplications are accomplished. Tiling both input and weight matrices, meaning the splitting of a large matrix into several smaller tiles, enables the processing of arbitrary matrix calculations.

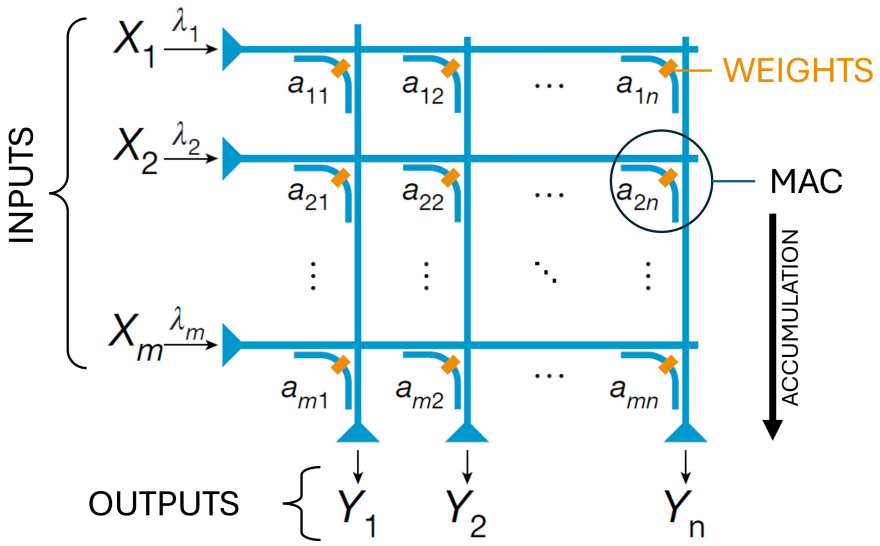
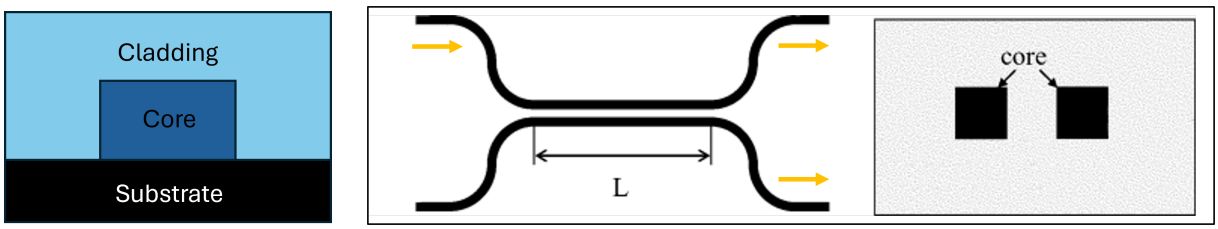


Figure 2.1.: Schematic depiction of a photonic $m \times n$ crossbar array. At each intersection i, j, an incoming light signal X_i is modulated with a factor $a_{i,j}$ and then accumulated over the column. The output of one column corresponds to an m-dimensional dot product, making the n-dimensional ensemble of outputs the result of a $(1 \times m) \cdot (m \times n)$ matrix-vector multiplication. Adapted from [14].

These MVMs will serve as the basis for artificial neural network operations.

Emission, Guiding and Coupling of Light

The light for a photonic circuit can be produced either on- or off-chip; however, the former poses challenges because of strong temperature sensitivity, design difficulties, and decreased experimental flexibility. Both the originally proposed tensor core and the one investigated in this work rely on external light sources for these reasons [14]. While it was once a common belief that coherence is a beneficial or even necessary characteristic of the light beam, recent studies by Brückerhoff-Plückelmann et al. have found that partially coherent signals can improve the performance and robustness of photonic tensor cores by allowing for higher and more straightforward parallelization [10]. That is why a partially coherent, broadband amplified spontaneous emission (ASE) source is utilized. In short, this device pumps a laser gain medium, creates a population inversion, and thereby amplifies spontaneous emission by stimulated emission. Additional intensification can be achieved by deploying a subsequent optical amplifier, like an erbium-doped fiber amplifier (EDFA), which operates in a similar manner. A drawback of this setup is the considerable introduction of noise, for example, in the form of intensity fluctuations. However, this noise is mostly Gaussian and can be mitigated by averaging the measurements taken with the PIC [30]. The software interfaces of the integrated system at hand, which this work is based on, already include a method to automatically apply averaging.



(a) On-chip waveguide **(b)** Directional coupler. Adapted from [32].

Figure 2.2.: Basic components of integrated photonic circuits. a) shows a schematic cross-sectional view of a waveguide, where the core acts as the transmission medium of a light ray. b) visualizes the function of a simple directional coupler as it is used in the analyzed photonic tensor core: Light of an initial intensity inside the upper waveguide couples evanescently into the lower waveguide, resulting in a signal in both outputs. Like this, the coupler is used as a power splitter. By adjusting the directional coupler, such as its coupling length L or the distance between the waveguides, the splitting ratio can be altered.

The directed transmission of light is achieved by optical fibers in free space or by waveguides on-chip. The latter constitute the basis of all integrated photonics and direct the light based on total internal reflection, just like their fiber counterparts in telecommunication. To confine a light ray, a core with a high refractive index n_{core}, for example made out of Si, acts as the main transmission medium, while it is surrounded by a cladding with lower refractive index n_{clad}, e.g. SiO_2 [21]. For total internal reflection to take effect, the light has to hit the boundary surface between the core and the cladding at a sufficiently large angle. Based on Snell's law, we can find an estimate for the critical angle; however, a detailed analysis of the behaviour of an electromagnetic wave inside a waveguide or optical fiber requires an investigation with the Maxwell equations [21].

For on-chip waveguides, the core material is typically deposited as a thin film on a substrate, then etched into the shape of a patterned resist mask, and finally clad [19]. This results in waveguides with a rectangular cross-sectional area, visualized in Figure 2.2a. The PIC investigated in this work was fabricated on a silicon-on-insulator (SOI) wafer, with the upper layer, composed of silicon, used as the waveguides' cores. The resist mask is applied by electron beam lithography, which can create very fine and precise structures, allowing for sizes of 0.1 µm and less [19]. The substrate below the silicon, as well as the cladding, consists of SiO_2.

The transition of light from an optical fiber (or a free-space beam) into an integrated waveguide is achieved by a grating coupler. This device harnesses the Bragg effect, diffracting incoming waves by a periodically spaced grating, which can then be directed into a horizontal waveguide. These couplers can achieve a maximum efficiency of 70-80%, making them an essential element of many modern PIC applications [21]. In the photonic

system of this work, the grating coupler enables a fiber array to be glued from above onto the integrated circuit, thus providing external light with great consistency and robustness.

The structure of the crossbar array requires multiple optical signals, one input signal for every intersection. The accumulation of the single MACs can only work as intended if the input power is equally divided. For this purpose, so-called directional couplers can be used. Inside these couplers, two waveguides are brought together so closely that the evanescent field of an electromagnetic wave—an oscillation extending beyond the core boundaries with exponential decay—in one waveguide overlaps into the other waveguide. This can lead to a power transfer [21]. Figure 2.2b visualizes the concept of a directional coupler. The coupling strength can be adjusted by adapting the geometry and materials of the device, but there is also a strong wavelength dependence [11]. The repeated dividing of the input intensity for the PIC leads to increased demands regarding the external light supply infrastructure, which emphasizes the need for an optical fiber amplifier following the ASE source.

Electro-Absorption Modulators

There exist different approaches to encoding target values in the input signal and the weights within the crossbar array. For their system, Feldmann et al. utilized variable optical attenuators to modulate the incoming rays and passive phase-change material (PCM) to set non-volatile weights [14]. The PCM can change its refractive index according to its physical structural state: The amorphous state has a lower index, while the crystalline (ordered) state has a higher one. By thermally controlling the optical properties of this material, which is placed on top of a waveguide, the amplitude or phase of the light can be tuned.

Our setup, in contrast, accomplishes both, imprinting values on the inputs and the weights, through the incorporation of electro-absorption modulators (EAMs). These are active semiconductors whose absorption coefficient increases with applied voltage [11]. They are responsible for nonlinear scaling behavior, which this work aims to mitigate with new control algorithms. To gain an idea of the reason for this non-ideality, the basic physical principles behind EAMs are briefly reviewed here.
The original design is based on the Franz-Keldysh effect: An electric field E tilts the electronic band structure, effectively modifying the optical absorption edge, as shown in Figure 2.3. The wave function of an electron near the top of the valence band with energy W extends into the band gap, decaying exponentially according to

$$\psi = u(r)e^{ikx}, \tag{2.2}$$

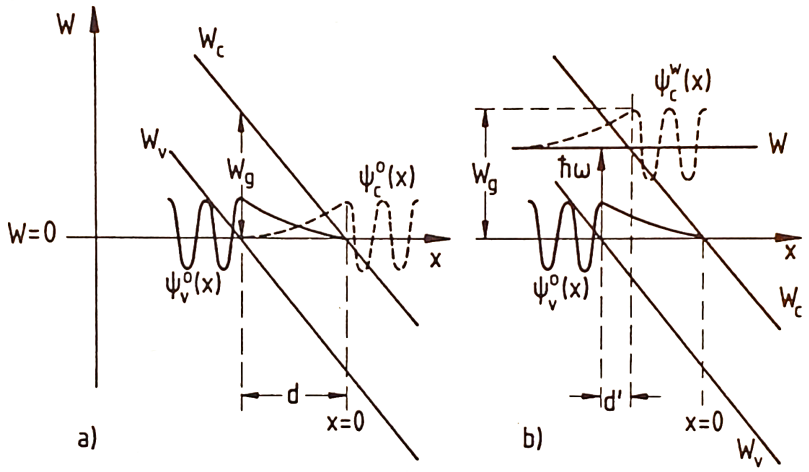


Figure 2.3.: Franz-Keldysh effect. The graphic shows an electron tunnel between the tilted valence and conduction bands when an electrical field E is applied in x-direction. In a), the electron energy is not altered, while b) illustrates the energy jump caused by a photon absorption. The schematic depiction of the electron wave functions ψ indicates a reduced tunnel distance for the absorption case. Taken from [11].

where k is an imaginary wavevector [11]. For an interband transition to occur, the electron has to tunnel across a triangular barrier, whose height is given by the band gap energy W_g and whose width d depends on the electric field:

$$d = \frac{W_g}{qE} \tag{2.3}$$

[11]. As evident, stronger electric fields reduce the tunnelling distance. If now a photon with energy $\hbar\omega < W_g$ is absorbed, the barrier decreases to

$$d' = \frac{W_g - \hbar\omega}{qE}, \tag{2.4}$$

making the tunnel process more likely. Conversely, the tilting of the bands and this field-assisted tunneling also increases the absorption probability for photons that would normally not have sufficient energy to lift an electron from the valence band into the conduction band. Since the tilting is controlled by the applied field, this provides a mechanism to modulate the optical absorption of an EAM using an external voltage [11].

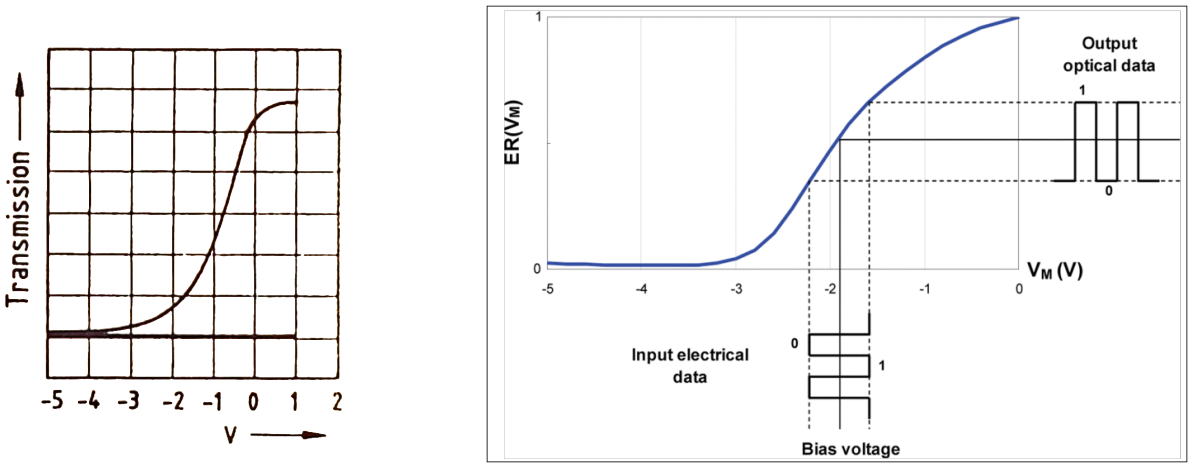
Modern EAMs rely on the quantum-confined Stark effect, which also describes the modification of band structure in response to electric fields; however, their mechanisms differ slightly and will not be discussed here. The current state-of-the-art EAMs can achieve extremely fast switching rates of several tens of Gigahertz, making them a suitable modulation device for high-throughput applications [21].

To characterize an EAM, a standard metric is the optical power extinction ratio (ER) as

a function of the applied bias voltage V_m. This is the relation of the output power P_{out} for a given voltage and the reference power without an electric field:

$$ER(V) = \frac{P_{\text{out}}(V = V_m)}{P_{\text{out}}(V = 0)} \tag{2.5}$$

[9]. A large extinction ratio corresponds to a strong transmission, and a small ratio reflects strong absorption. Figure 2.4 shows the typical absorption behavior of EAMs, where we can observe an S-shaped graph. This non-linear relationship complicates the modulation and has to be dealt with during the weight setting process.



(a) Qualitative transmission **(b)** Extinction ratio curve of a modern EAM

Figure 2.4.: Absorption behavior of EAMs. a) illustrates the qualitative transmission curve of an AlGaAs-EAM, while b) plots the normalized optical extinction ratio of an EAM at 1550 nm. In both figures, we observe non-linear scaling of the absorption. In b) we can also see how an electrical input signal can be modulated onto an optical one. Note, however, that this represents a digital signal, whereas in this work, analog data with continuous intensity levels will be processed. Taken from [11] and [9] respectively.

Additional Components

To constitute a working system with the integrated photonic circuit, some more components are relevant, which will be briefly mentioned here: First of all, the EAMs are controlled by analog voltage signals, but the input for the MVMs is present in digital form. Thus, a conversion via a digital-to-analog converter (DAC) is necessary. After light is processed inside the crossbar array and accumulated, it needs to be measured to obtain the calculation result. The analyzed PIC is equipped with embedded photodiodes for that purpose, which produce a photocurrent in response to incoming light. This current then needs to be amplified and turned into a voltage for easier readout, which is achieved by transimpedance amplifiers (TIA). Finally, the computation outcome remains in analog form, which is why the counterpart of the DAC, an analog-to-digital converter (ADC),

is utilized to enable evaluation and further processing using conventional von-Neumann computer architectures. The integration and operation of all these supplementary elements pose a challenge by themselves and require an elaborate electronic infrastructure and meticulous coordination.
Chapter 3 will provide a more detailed explanation of the working and methodology of the setup investigated in this study.

Wavelength Division Multiplexing

One significant advantage of optical processing in terms of parallelization is the ability to scale within the frequency space: By encoding information in the wavelength of the carrier light, multiple inputs can be processed simultaneously. The key technique to achieve this is wavelength division multiplexing (WDM), which has already been demonstrated in the original publication [14]. It can be realized, for instance, by generating an optical frequency comb—a coherent spectrum made up of discrete and regularly spaced spectral lines [17]—and using different frequencies that do not interfere with each other to process different inputs inside a photonic tensor core [14]. While the current experimental setup is used without WDM, this poses a great opportunity in future buildups to massively parallelize computations.

2.1.2. Nonidealities of Analog Photonic Computing

As seen in the previous paragraphs, operating a photonic tensor core involves controlling multiple electronic, opto-electronic, and optical components and coordinating signals in digital, analog electrical, and analog photonic forms. Within this chain, potentially undesirable behavior can occur, which we refer to as non-idealities. When working with PICs and their support structure, these effects must be taken into account to fully leverage the potential of a hardware accelerator. Algorithms introduced in chapter 3 will address some non-idealities, especially those involved in the weight setting process. In detail, three phenomena are considered in this work:

- non-linear weight scaling
- inhomogenities between optical channels
- electrical crosstalk

The first aspect was already mentioned when discussing electro-absorption modulators. The fundamental relationship between the applied voltage and the absorption coefficient in these semiconductors is not linear. Although mostly exploiting the approximately linear region of the ER-curve shown in Figure 2.4, nonlinear scaling is observed around the

boundaries of their operating range, which is especially pronounced when transmitting analog instead of digital samples because of the use of a continuous range of voltages rather than two distinct values [16].
Inhomogenities between optical channels, e.g., as a consequence of fabrication faults, are reflected by differing output response ranges when measuring the photo stream with the photodiodes. In detail, this means that the photocurrents of some channels vary, despite receiving the same input power and being set with exactly the same weights.
Finally, crosstalk is not an exclusive problem of photonic chips, but rather of circuitry in general. This broad term refers to the unwanted influence among seemingly independent signals, caused by capacitive or inductive effects [24]. In the present case, this materializes as the input and weights of one channel having a slight impact on the output in a separate channel.

In this study, these non-idealities will be primarily regarded as inherent hardware conditions. Nevertheless, they must be accounted for to obtain optimal experimental results. As a consequence, pre- and post-processing schemes were developed to mitigate them.

2.2. Machine Learning Basics

The following part gives an overview of the fundamental principles of neural networks and demonstrates how their internal functioning primarily involves matrix-vector multiplications, which can be efficiently performed using an optical hardware accelerator. Furthermore, the specific architecture of convolutional neural networks (CNNs) is explained, along with how this type of machine learning excels at computer vision tasks, such as image recognition, which is a suitable use case for photonic crossbar arrays.

2.2.1. Artificial Neural Networks

To understand artificial neural networks (ANNs), it is useful to first examine one of the earliest implementations of a Machine Learning algorithm: the famous *perceptron* developed by Frank Rosenblatt in 1958 [15]. It is a model of a single neuron, with several weighted inputs and one output, that can perform binary classification. Given an input vector x and a weight vector w, the perceptron can be modeled as

$$\hat{y} = \varphi\left(\sum_i x_i w_i + b\right), \tag{2.6}$$

with $\hat{y}$ being the output of the neuron, an offset value b called bias, and a non-linear activation function φ, such as a sigmoid function or a rectified linear unit (ReLU) [29].

The elements of x are called features, and in a concatenation of several inputs, each ensemble of features is referred to as one observation or sample. To classify P observations with N features each, the linear neuron part calculates the matrix-vector multiplication $X^T w$, such that the perceptron predicts a value y_p for every sample:

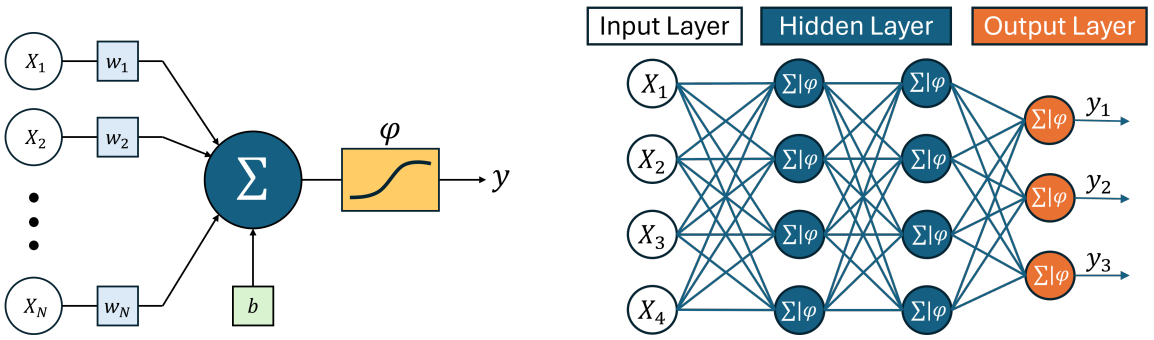
$$\hat{y} = \varphi(X^T w + b) = \varphi\left[\begin{pmatrix} X_{1,1} & X_{1,2} & \dots & X_{1,N} \\ X_{2,1} & X_{2,2} & \dots & X_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ X_{P,1} & X_{P,2} & \dots & X_{P,N} \end{pmatrix} \cdot \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix} + \begin{pmatrix} b \\ b \\ \vdots \\ b \end{pmatrix}\right] = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_P \end{pmatrix} \tag{2.7}$$

Here, we follow the convention of $X \in \mathbb{R}^{N \times P}$ [31]. An illustration of this process is given in Figure 2.5a. In compact notation, the bias is included inside the matrix-vector multiplication by defining $\tilde{X}_p^T = \left[1, X_p^T\right]$ and $\tilde{w} = [b, w]^T$, simplifying the equation to $\hat{y} = \varphi(\tilde{X}^T \tilde{w})$ [31]. This mathematical model consists of $N + 1$ parameters from the weights and the bias that can be tuned to transform the predicted distribution $\hat{y}$ of a given weight initialization towards a target distribution y. In terms of classification, the y are labels associated with the input data, in the original perceptron 0 or 1, that the neuron is supposed to predict. This parameter adjustment is called supervised learning. It is achieved by defining a scalar loss function $\mathcal{L}(y, \hat{y}) = \mathcal{L}(y, \tilde{X}; \tilde{w})$ that quantifies the deviation between the prediction and the target, and then finding the optimal parameters by optimizing the loss. In practice, this happens iteratively by repeatedly predicting all samples and moving the weights in the direction of the negative gradient at each time step:

$$\begin{aligned} \tilde{w}^{t+1} &= \tilde{w}^t - \alpha g^t \\ g^t &= \nabla_{\tilde{w}} \sum_{i=1}^{P} \mathcal{L}(y_i, \tilde{X}_i; \tilde{w}^t) \end{aligned} \tag{2.8}$$

This method is known as gradient descent, with the hyperparameter α representing the learning rate [31]. More elaborate optimization schemes exist, such as stochastic gradient descent (SGD) or ADAM, that offer faster and better generalizing training in more complex scenarios. Details about their principles can be found, for example, in [12].

To perform multiclass predictions, several artificial neurons are required in a vertical stack, each fed by the same input data but with distinct weights and biases $\tilde{w}_c$. Each perceptron is then responsible for recognizing one of C classes, meaning that the outputs and weights are now grouped in matrices $\hat{Y} \in \mathbb{R}^{P \times C}$ and $W \in \mathbb{R}^{N \times C}$, respectively. To determine the single most likely class, argmax is applied on each sample output, resulting in a one-hot encoded vector. If we want to get an estimate of the probabilities for all



(a) Artificial neuron **(b)** Multilayer perceptron

Figure 2.5.: Basic building blocks of artificial neural networks. Part a) visualizes the famous perceptron model, a strongly simplified imitation of a biological neuron. Inputs X_n are weighted with w_n and summed up, together with the bias b. The result gets transformed by a nonlinear activation function φ. In b), a concatenation of neurons into a network can be seen, also known as a multilayer perceptron. Each line between two layers represents a weight. The output can be fed into a softmax function to yield the probabilities for each class.

classes, the standard method is to use the soft(arg)max formula:

$$\mu_{p,c} = \frac{\exp(\hat{y}_{p,c})}{\sum_{i=1}^{C} \exp(\hat{y}_{p,i})} \tag{2.9}$$

The $\mu_{p,c}$ reflect the probability of a sample p belonging to class c, fulfilling the condition $\sum_{i=1}^{C} \mu_{p,i} = 1$ [31]. A common loss function for multiclass scenarios is the cross-entropy loss

$$\mathcal{L}(y, \tilde{X}; \tilde{w}) = -\sum_{i=1}^{C} y_i \log(\hat{y}_i) = -\sum_{i=1}^{C} y_i \log(\varphi(\tilde{X}\tilde{w}_i) \tag{2.10}$$

[31], which will also be deployed in machine learning algorithms investigated in this work.

A multilayer perceptron (MLP), the basic structure of an ANN, is a horizontal ensemble of several of these vertical neuron stacks, illustrated in Figure 2.5b. It consists of an input layer, an output layer with C neurons, and intermediate layers of various sizes, also called hidden layers. When all neurons in one layer are connected to all neurons in the following one, the layer is referred to as fully connected (FC). As an example, the output of an MLP with two hidden layers and one output layer can be expressed as

$$\begin{aligned} \hat{Y} &= \varphi(X_2^T W_2 + B_2) = \varphi(\varphi(X_1^T W_1 + B_1) \cdot W_2 + B_2) \\ &= \varphi(\varphi(\varphi(X_0^T W_0 + B_0) \cdot W_1 + B_1) \cdot W_2 + B_2), \end{aligned} \tag{2.11}$$

This emphasizes that the fundamental operations of a neural network are matrix multiplications, which we aim to accelerate with photonic hardware.

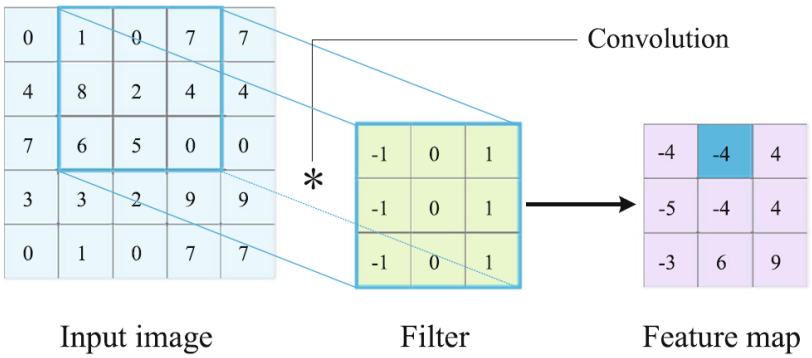


Figure 2.6.: Concept of a discrete two-dimensional convolution. A filter (or kernel) is slid over an input image, and at each position, the covered pixel values are multiplied by the filter weights and summed up to result in a single scalar output. All outputs form a feature map. Taken from [33].

2.2.2. Convolutional Neural Networks

A special architecture of ANNs is the convolutional neural network (CNN). It incorporates the discrete convolution operation to process data, especially images, and learn to extract features such as edges, textures, and more complex shapes. CNNs have proven superior to classical MLPs in the context of image recognition, for instance, and play key roles in modern computer vision applications [33]. The underlying concept of discrete convolutions in the two-dimensional case is visualized in Figure 2.6: A kernel, usually of size 3×3 or 5×5 is moved over an input image and at each position, the overlapping pixel values and filter weights are multiplied and summed, which results in a feature map that can then be further processed [33]. From a mathematical perspective, a convolution is a special form of an MVM, which is why it can be processed optically on PICs. Indeed, convolutions are particularly well-suited operations for photonic tensor cores because they compute vast amounts of input data with relatively few weights, thus optimally leveraging the inner workings of the optical hardware [14]. More details on that matter are outlined in Section 3.1.

Inside a convolutional layer of a CNN, the weights of the kernel are learnable parameters, and multiple kernels are applied to identify distinct features. To perform classification, the outputs of several convolutional layers are typically further processed by subsequent FC layers. To reduce the data dimensionality within the network, pooling layers are incorporated. This divides a given image or feature map into separate regions, e.g., tiles of size 2×2, from which a single value is extracted. A popular method is max pooling, in which the maximum value from each region is drawn [33]. This down-sampling enables the use of more kernels in subsequent layers, which is why CNNs commonly increase their

channel width in deeper parts of their structure, as seen, for example, in the famous VGG architecture introduced in 2014 [26]. In this work, the designed convolutional neural networks will follow this concept.

The actual implementations of MLPs and CNNs include many additional aspects whose explanation would exceed the scope of this thesis. It can be summarized, though, that both fundamentally rely on MAC operations, which can be performed using the photonic crossbar array. Chapter 3 will describe how exactly the hardware at hand can be used to deploy multi-layer neural networks in the optical domain.

3. Methods and Experimental Setup

In this work, experiments were conducted on an existing integrated and fully operational photonic system[1]. The first Section (3.1) aims to illuminate the fundamental working of the setup, including a more detailed explanation of the hardware components and the existing software interface. The subsequent Section (3.2) will then present control algorithms created by the author to enhance the system's operation and to perform machine learning applications with the photonic accelerator. Experiments evaluating the new functions and the system's processing characteristics are also described. The third part of this chapter, presented in (3.3), addresses the photonic machine learning studies. It introduces known methods of hardware-aware training for neural networks and outlines the methodology used in the image recognition tests conducted on the optical hardware.

3.1. The Integrated Photonic System

3.1.1. Photonic Integrated Circuit

The core of the experimental setup is the photonic integrated circuit (PIC), which is equipped with the input grating couplers, the electro-absorption modulators (EAMs) to set the values of the inputs and the weights, the optical crossbar array in which the photonic MAC operations happen, and photodiodes which detect the light and transform it into a photocurrent. Figure 3.1 shows these elements on the PIC.

The crossbar array is of size 9×3, but in the current configuration, only six out of nine channels are used. As explained in the theory Section 2.1, the encoding of values is realized by differences in the transmission of the incoming light through the EAMs. The light intensity is a physical quantity and, by its nature, non-negative, not allowing for negative weights immediately. Certain applications can operate effectively using only non-negative values, for instance, some machine learning tasks including image recognition [8]. However, a study by Becker et al. has shown that these restrictions also noticeably limit the maximum accuracy for more complex image recognition benchmarks, such as CIFAR-10 [3]. To avoid being constrained by positive-only weights, a method called

[1]The entire system, including both hardware and software interfaces, was developed by Jelle Dijkstra, Neuromorphic Quantumphotonics, Heidelberg University

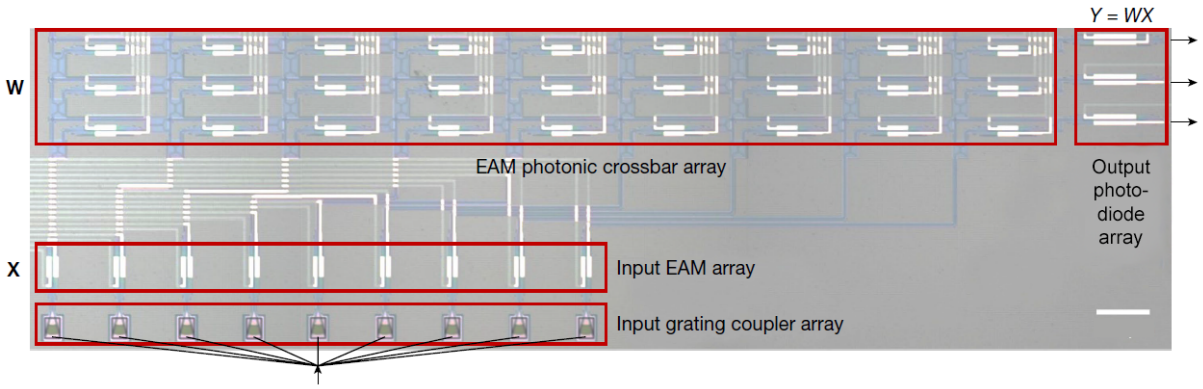


Figure 3.1.: Image of the PIC tensor core. From the crossbar array of size 9×3, a 6×2 subset is used, resulting in six input channels and twelve matrix weights, all controlled by EAMs. Light is injected via the grating coupler array. Balanced readout enables the processing of negative weights despite the non-negative nature of the optical intensity, resulting in an effective capability of processing six MAC operations per time step. Taken from [10].

balanced readout is employed, which utilizes one row of the crossbar array as a reference, allowing the difference between transmissions to be interpreted as a signed value. Here, the subtraction is carried out in the FPGA. Given a physical weight w^p_{max} associated with the maximum transmission level and accordingly a minimum, yet nonzero, physical weight w^p_{min}, the neutral weight is defined as the arithmetic mean:

$$w^p_{\text{ref}} = \frac{w^p_{\text{max}} + w^p_{\text{min}}}{2}. \tag{3.1}$$

With the range of physical weights given by $\Delta w^p = w^p_{\text{max}} - w^p_{\text{min}}$, the implemented algorithm encodes an input weight $w \in [-1, 1]$ in the subtraction of two physical weights w^p_1 and w^p_2 as follows:

$$\begin{aligned} w &= \frac{w^p_1 - w^p_2}{\Delta w^p} \\ \text{with} \quad w^p_1 &= w^p_{\text{ref}} + w\frac{\Delta w^p}{2}, \quad w^p_2 = w^p_{\text{ref}} - w\frac{\Delta w^p}{2}. \end{aligned} \tag{3.2}$$

This means that the weight w^p_2, associated with the reference row, is symmetrically offset from the neutral level w^p_{ref}, which acts as the zero point. As a consequence, the difference between the photocurrents of the main and the reference rows spans a range proportional to $[-\Delta w^p, +\Delta w^p]$.

Due to the balanced readout, only two rows of the crossbar array are utilized, resulting in twelve MAC operations that correspond to six operations with both positive and negative weights. This means that the calculations per time step are essentially a six-dimensional

dot product of an input vector X and a weight vector w:

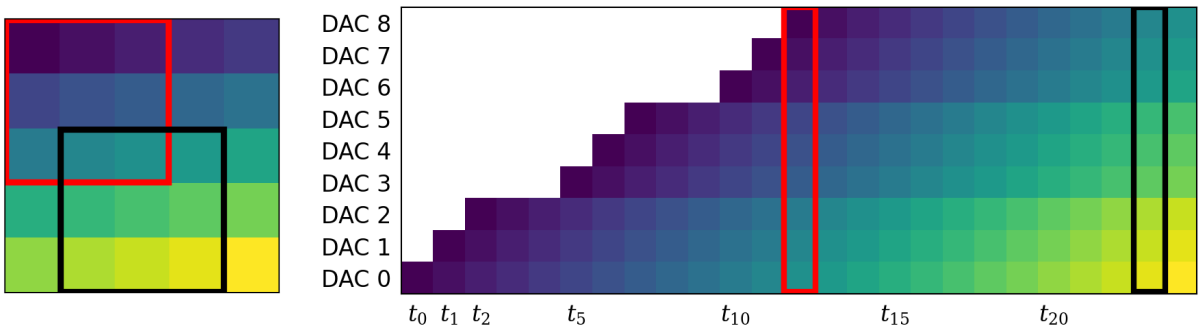
$$Xw = \sum_{i}^{c} X_i w_i = \sum_{i}^{c} X_i \frac{w_{1,i}^{p} - w_{2,i}^{p}}{\Delta w^{p}} \tag{3.3}$$

As the integrated system is still a prototype and in development, the third row is currently not in use. In general, balanced readout schemes can also work with a single reference channel for multiple calculation channels [10], but this reduces the maximum accessible photocurrent range by half. While future photonic tensor cores will likely be larger, the challenge of this work will be to perform meaningful calculations already with this small usable crossbar array. This will serve as a preview of what can be achieved when scaling the optical structures.

3.1.2. Electronic Interface

The PIC itself only contains (electro-) optical components and thus requires a specific electronic infrastructure to function properly. This includes the digital-to-analog converters (DACs) that control the EAMs, the analog-to-digital converters (ADCs), and the support system to operate these devices based on abstract digital inputs. Furthermore, the photocurrent measured by the photodiodes on the PIC must first be amplified and translated into a voltage by transimpedance amplifiers (TIAs) before the ADCs can yield the processing results in digital form. Regarding the DACs, two different types are utilized in the system at hand: 16 high-speed radio frequency DACs (RF-DACs) that transmit modulation values to the input EAMs, and slower multi-channel DACs that set the weights inside the crossbar array.
The RF-DACs, as well as the ADCs (also high-speed RF-ADCs), are integrated on a so-called radio frequency System-on-Chip (RFSoC), which accounts for the control of the entire setup. A central processing unit (CPU) on this RFSoC runs a Linux operating system that can host Python servers, thus being responsible for receiving external commands and data. Besides this conventional processing unit, the RFSoC contains a field-programmable gate array (FPGA), which is specifically adjusted for the given task and transforms the data from the CPU into suitable signals for the DACs. These elements are embedded onto a printed circuit board (PCB), the RFSoC evaluation board, which provides additional support infrastructure such as power supply and connectors. The model specifications can be found in [1]. The TIAs and the weight DACs are not integrated on the RFSoC board, but they are directly connected to it.

At the ground level, the FPGA is programmed to perform a convolution. This has practical advantages, but is also suitable for photonic computations in general: A convolution is an operation with potentially large input matrices but rather small and especially static



(a) Example image **(b)** Corresponding DAC inputs

Figure 3.2.: Visualization of convolution processed inside the FPGA. Given the example image a) as input, the FPGA sends the respective values to the different RF-DACs according to b). The red and the black rectangles symbolize two kernel positions in the space domain of the image and the time domain of the analog signal. The white pixels represent zeros, which are a consequence of a special padding scheme. Note that the padding occurs only at the very beginning, until the kernel is fully aligned with the data. Afterward, the kernel jumps column by column from the right end of the image to the left end. This happens because the image is internally flattened and seen as a 1D array. The process can be imagined as sliding the kernel with its anchor point in the lower right corner (input of DAC 0) over each input pixel, resulting in N analog signals per DAC for N pixels.

weight matrices. The PIC allows for extremely high bandwidths for input data, but controlling the weights is slower due to significantly larger data rates. An application that processes vast amounts of input data but requires relatively rare switching of weights optimally leverages the potential of the photonic accelerator, especially with highly parallelized computations enabled by wavelength-division multiplexing (WDM) [14]. On the practical side, convolutions are also convenient, for example, because an input data stream causes an output stream of the same size, such that adjustment methods can be performed more easily.

With convolutions being the primary target application, the FPGA's operation is highly efficient, as it is optimized for these specialized MVMs rather than general ones. It would also be possible to calculate a convolution if the FPGA were fundamentally designed to perform general MVMs, but this would induce sending image data several times from the memory to the programmable logic (compare also Section 3.2.2). Since each pixel of an image is used multiple times during a convolution, duplication of the data must occur at some point. However, in the realized implementation, this duplication happens at the last possible step within the pipeline, namely in the FPGA. Figure 3.2 illustrates the convolution process at a hardware level, showing how an input image is transformed within the FPGA and then sent to the digital-to-analog converters.

Nonetheless, it is also desirable to be able to compute general MVMs, for example, for fully connected layers of neural networks. Given this internal hardware function, a task of

this work is to develop or refine suitable control algorithms that perform both convolutions and conventional matrix-vector multiplications. Through Figure 3.2, it is apparent that the internal working is laid out for nine channels, which means that the control algorithms have to account for only six channels being used. Section 3.2 will focus on these algorithms.

3.1.3. General Setup

The whole experimental setup comprises an integrated system that combines the photonic integrated circuit with the hardware interface of the RFSoC board. Together, these components are capable of handling all tasks, including modulation, processing, and detection of light, as well as digital-analog and analog-digital conversions. The light necessary to operate this setup is provided by an external source, specifically an amplified spontaneous emission (ASE) source in conjunction with an optical amplifier. Via an Ethernet port on the evaluation board, input data and control signals can be transmitted from a computer. The whole workflow is schematically presented in Figure 3.3.

As mentioned in the theoretical background, the ASE source and the optical amplifier introduce uncorrelated noise to the setup, for example, in the form of intensity and phase fluctuations, which are reflected in the photocurrent. However, this can be mitigated by repeating measurements and averaging them, increasing the signal-to-noise ratio. While the light source has been fixed, preliminary tests were conducted with different amplifier arrangements, including an erbium-doped fiber amplifier (EDFA) and low-noise high-power optical fiber pre-amplifiers. The deployment of the single EDFA has proven to yield the most consistent results, which is why it was chosen for all subsequent experiments with the optical setup.

Processing Speed

The input processing speed of the system is primarily determined by the electrical components, such as the DACs, rather than the photonic ones. For technical reasons, the maximum input frequency of the FPGA is set to 4 GSas^{-1}. However, because only AC signals can be sent, not every sample can encode a single symbol, which is why a modulation is required: Two modulation schemes can be used with the system; they are illustrated in Figure 3.4. Both encode a single symbol with four samples, reducing the data rate to 1 GSas^{-1}. Furthermore, sending each symbol several times significantly improves the accuracy and reduces effects caused by impedance mismatching, which is why experiments are carried out with an upsampling setting of 16. Therefore, the real operation throughput is 62.5 MSas^{-1}. It must be further taken into account that repeated measuring to reduce uncorrelated noise takes up time resources as well.

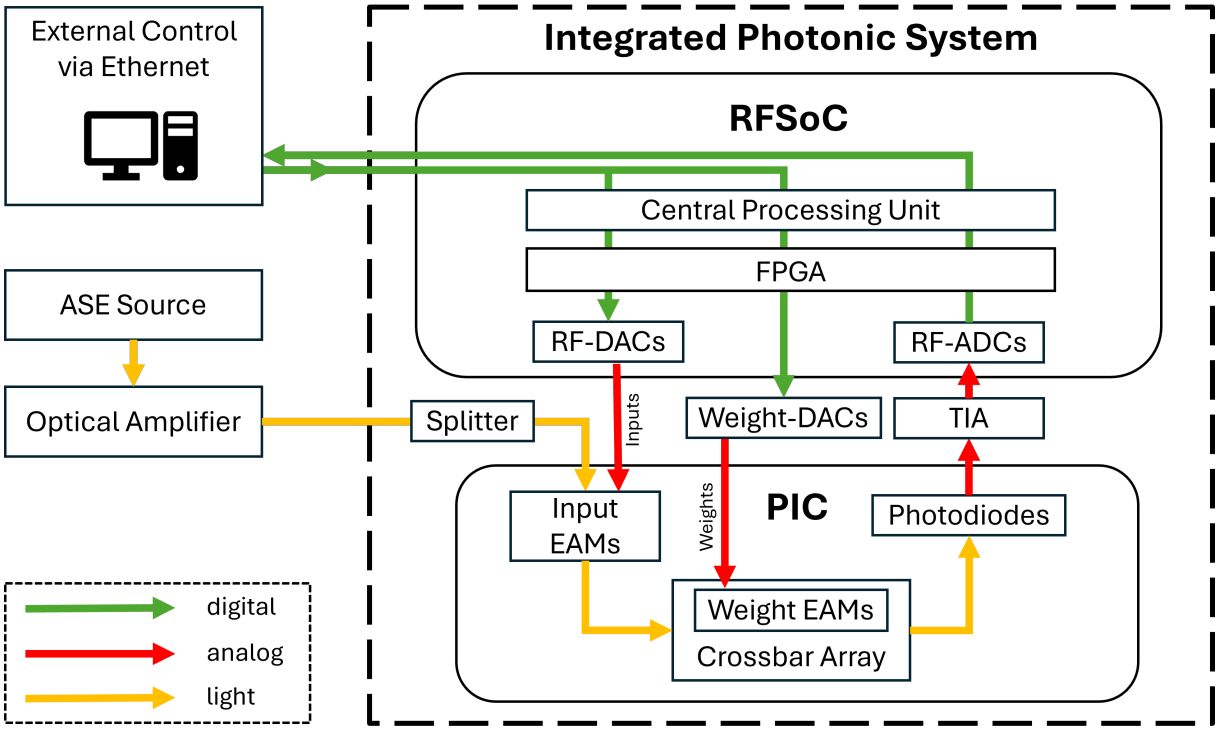


Figure 3.3.: Schematic working of the experimental setup. Data is transmitted digitally via Ethernet to the radio frequency System-on-Chip (RFSoC), where it is processed by the CPU and forwarded to the field-programmable gate array (FPGA). The FPGA then sends the conditioned data to the digital-to-analog converters (DACs), the inputs to the radio frequency DACs, and the weights to the slower DACs. The obtained analog signal is used to control the high-speed electro-absorption modulators (EAMs) within the photonic integrated circuit (PIC), thereby setting the input modulation and the weights within the crossbar array. Light is produced externally via an ASE source, then amplified and split evenly towards the input channels of the tensor core. At the interface of the PIC, the light gets coupled from a glued fiber array inside the waveguides by a grating coupler. Within the crossbar array, directional couplers divide the power equally, allowing MAC operations to be performed. On-chip photodiodes measure the resulting light stream. The photocurrent is then amplified by a transimpedance amplifier (TIA) and transformed into a digital signal by the analog-to-digital converter (ADCs).

In theory, the weights could be set at the same speed as the inputs, as the high-speed EAMs also control them. However, this would also result in tremendous data rates, as the number of required samples per switching is the number of samples sent for each input multiplied by the depth of the crossbar array, thus it scales rapidly for larger tensor cores. Furthermore, the electronic interface is not equipped with enough RF-DACs, which is why the setup makes use of the slower multi-channel DACs and enables weight switching within the kilohertz region.

As a result, the system, in its current configuration, does not yet offer an advantage over traditional CPUs or graphics processing units (GPUs) in terms of computation time. However, these speed limitations posed by the electronic support system of the photonic tensor core are not fundamental and can be expected to become less relevant in future

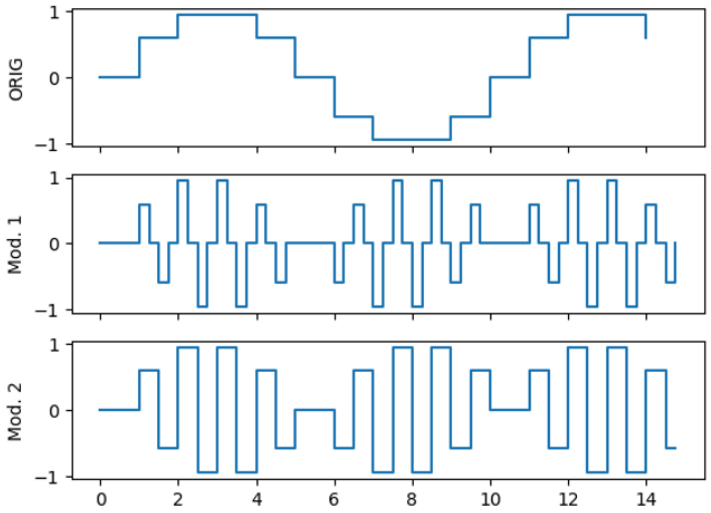


Figure 3.4.: Modulation Schemes. A single symbol is encoded into four samples. Method 2 (bottom) sends two samples of the desired amplitude and then two samples with swapped signs, while method 1 (middle) inserts zero samples in between. See [2].

integrated setups. The photonic chip itself is capable of processing at least 30 GSas^{-1} [10] and offers prospects for future computational performance. The methods developed in this work are universal and can be implemented in subsequent PICs.

3.1.4. Software Interface

External access to the photonic integrated system is achieved via a Jupyter server running on the RFSoC. An extensive Python library operates the RFSoC—thus the whole hardware with the exception of the light source and its amplifier—and provides a high-level software interface. The work described in the subsequent sections builds upon this interface. The most critical functions include a method to set relative weights $w \in [-1, 1]$ of the channels (accounting automatically for the balanced readout), a method to measure the current response of the photodiodes (explained in the next section), and - most importantly - an automatic function to processes a given array of data on the PIC and to return the results according to the internal computation explained in 3.1.2. The latter handles preprocessing inside the programmable logic, the conversion from digital signals to analog ones, thereby the respective modulation on the photonic integrated circuit, the reconversion of the analog result signals, and finally the data synchronization between the RFSoCs' input and output streams. It also takes care of the averaging setting: Since the division of multiple measurements is realized by a bit shift within the FPGA, the averaging parameter can be set to a power of two. Many additional support functions of this library have been utilized, but their details are not relevant for the understanding of this work. It can be summarized that the existing software interface allows for abstract coding, as the hardware coordination is automatically handled.

3.2. New Control Algorithms

As seen in the theory chapter, deep learning basically performs matrix multiplications to simulate the function of an artificial neuron and to forward signals between layers. To process these tasks on the given photonic accelerator, specific control algorithms must be employed. This includes, especially, a reliable way to set the weights and scale the PIC's output, as well as the actual matrix multiplication function and a convolution algorithm that can be used for convolutional neural networks. While the previous section has explained the existing hardware and software interface, this section introduces new algorithms or modified versions of previous methods, created by the author.

3.2.1. Weight Setting Algorithm

One of the challenges of photonic or analog computing in general is the nonlinear scaling of weights and errors that occur during the setting process. The discussion in 2.1.2 has shown that this comes partly as a consequence of the fundamental EAM behavior, but also due to electronic imperfections in analog circuitry. A weight setting algorithm has to account for this. Since the actual result of an MVM operation is determined by a balanced readout between two rows in the crossbar array, just the relative proportions of the weights to each other matter. An algorithm thus only needs to focus on conserving the relativity of the weights, while the actual scale has no significance. The more accurate the relative weights are, the more accuracy we can expect from the subsequent calculations. Furthermore, time constraints play a key role in setting the weights: Artificial neural networks include huge numbers of distinct weights, even small nets have usually already ten thousands of weights, not to speak about deep convolutional neural networks or even transformers, which consist of millions to billions or nowadays trillions of parameters [5][13]. Therefore, machine learning applications require fast switching of weights. The two main metrics to evaluate a weight setting algorithm are, consequently, accuracy and time consumption. Included in the demand for high accuracy is the need for consistency and reliability.

Measuring the Set Weights

On the hardware side, the weight programming is not perfectly precise, which is why desired weights differ from the actually set ones. A function of the existing software interface enables the determination of these actual weights, in which a special MAC operation is applied and its outcome is measured. In detail, to test an exemplary channel

c_{test}, it receives an input of one, while zeros are sent to all other channels. The resulting scalar product thus only gives out the weight value $w_{c_{\text{test}}}$ of channel c_{test}:

$$\vec{X} \cdot \vec{w} = 1 \cdot \vec{e}_{c_{\text{test}}} \cdot \sum_i w_i \vec{e}_i = w_{c_{\text{test}}} \tag{3.4}$$

This operation is done in a sequence for all channels to obtain all real weights. The whole process takes about 65 ms and can be described as a measurement of the current input response.

There are different approaches to achieve the weight setting under the given circumstances. An iterative method had already been implemented and will be briefly explained. Afterwards, a new scheme is introduced that aims to improve the iterative algorithm drastically in both important metrics.

Iterative Method

The idea of the iterative algorithm is to incrementally shift an initial weight distribution towards the target one. This is achieved by adding a correction term to the relative weights that is proportional to the difference between the target and the actual weights. This very general methodology is capable of capturing nonlinearities and inhomogeneities among the channels by moving in small steps and constantly measuring the deviation. However, the repeated measure of the input response slows down the algorithm, which is a problem especially when it needs several dozen iterations to converge. Furthermore, the performance of this method is also dependent on the initial set of weights: Given that all channels observe a similar input response range, one can simply use a rescaled version of the desired weights as a starting distribution and expect a good estimate of the ideal distribution. But in cases where channels behave inhomogeneously, that initial guess might be much less effective.
A fundamental issue in this concept is that the program does not know anything about the underlying system; it cannot remember non-ideal behavior and goes through the same weight-shifting process over and over again. Of course, it would be thinkable to create a huge dictionary that saves many target weight combinations and their optimal relative weights, but that process scales exponentially with the number of available channels, thus it will be very time and memory-consuming. Another, much more promising alternative is to create a mapping of relative weights and their respective input response for each channel and calculate the optimal setting from that. A newly developed scheme will be presented below that follows this exact approach.

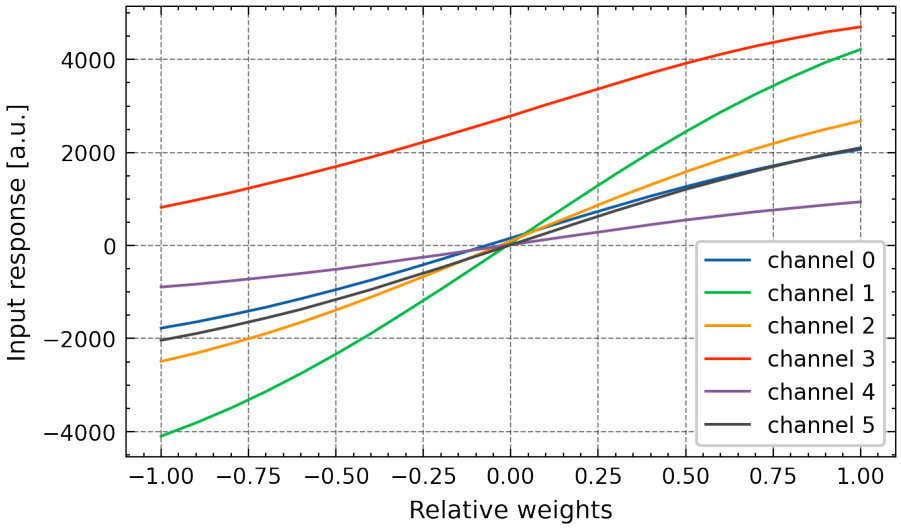


Figure 3.5.: Typical weight map. The measurement captures the input responses of each channel for the whole range of relative weights. The nonlinear scaling becomes obvious, resembling the EAM transmission curve shown in Figure 2.4, as well as the strongly inhomogeneous behavior between different channels. The red curve indicates a channel where the reference row in the crossbar array is not working; therefore, only positive weights can be set with that specific channel.

Creation of a Weight Mapping

The idea of the weight map is to determine which relative weight leads to which exact output on a single channel, such that taking the inverse yields a function that gives out the required relative weight given a desired outcome in that channel. When a mapping for all channels is obtained, one can simply set a target distribution in the unit of the input response range and receive the weight distribution that has to be actually set in the PIC. The prerequisite is that the scaling of each channel can be nonlinear but always monotonic, such that an inverse function can be found numerically. This underlying assumption is quite reasonable because if it were not true, more input power would not correspond to more output power, and the chip would malfunction.

To create the mapping, each channel's output response is measured for a distinct number of relative weights, which are equally distributed within the range $[-1, 1]$. All channels that are not tested receive a relative weight of zero. The mapping values in between the datapoints are interpolated linearly to obtain a continuous function. This process is repeated for all channels. Figure 3.5 depicts an exemplary measurement of a map, clearly showing the nonlinear weight scaling and inhomogeneities between channels.

As expected, the function between relative weights and their output response is monotonic, thus, the inverse can be calculated.
The duration of one map measurement depends on the number of data samples per channel and on the settings of the FPGA, such as the number of sample repeats. In the

standard setup, it takes (15.90 ± 0.03) s to measure 21 relative values for each channel. The goal is to complete this process only once, making it essentially an overhead calculation. To estimate the stability of the mapping, a measurement series was conducted that analyzed the deviation of the mapping from one initial state over the period of two hours. The results can be found in 4.1.

The map should already enable a precise setting of targets. However, this assumes that the channels are independent, which has turned out to be wrong. When measuring the input response of one channel, it turns out that the weight settings of other, usually neighboring channels, have a notable impact on the result. This so-called crosstalk is most likely of an electrical nature and happens on the PCB of the photonic chip. An effort has been made to eliminate or at least reduce the unwanted crosstalk; still, a certain amount remains and is a characteristic of the setup. To account for this effect, an algorithm has been programmed that automatically carries out a thorough crosstalk analysis and calculates linear corrections for a subsequent weight setting process.

Crosstalk Analysis

A complete measurement of all correlations would require a multidimensional grid search, scaling just as fast as a plain weight dictionary would. This is not feasible even for small crossbar arrays, so only relations between two channels are analyzed simultaneously. When investigating crosstalk to channel c_{test}, the algorithm measures its weight map several times for varying weight settings of another channel $c_{\text{reference}}$, while all remaining channels are set to zero. This scheme repeats such that every channel acts once as the reference. For n channels, we must do $n \cdot (n-1)$ map sweeps to quantify all bidirectional correlations. Since the current PIC is only used with six columns of its crossbar array, this is still a reasonable operation, taking about (14.582 ± 0.019) minutes with eleven reference values for each test pair $c_{\text{test}}, c_{\text{ref}}$. However, since the number of pairs grows quadratically with the number of channels, this may pose a significant computational overhead in larger photonic chips. In that case, it would be more efficient to measure correlations only between actual neighboring channels, resulting in a linear scaling. That this approach remains valid is demonstrated by Figure 3.6, which shows a typical representation of crosstalk relations between the channels. The largest correlations derive from physically adjacent connectors.

Given this data, linear functions without intercept are fitted for every set of test and reference channels. The slope of each fitted line is then used as a correction term when choosing the target weights. Figure 3.7 shows an exemplary linear fit between two channels. Note that this correction can only function if the crosstalk is a consequence of the

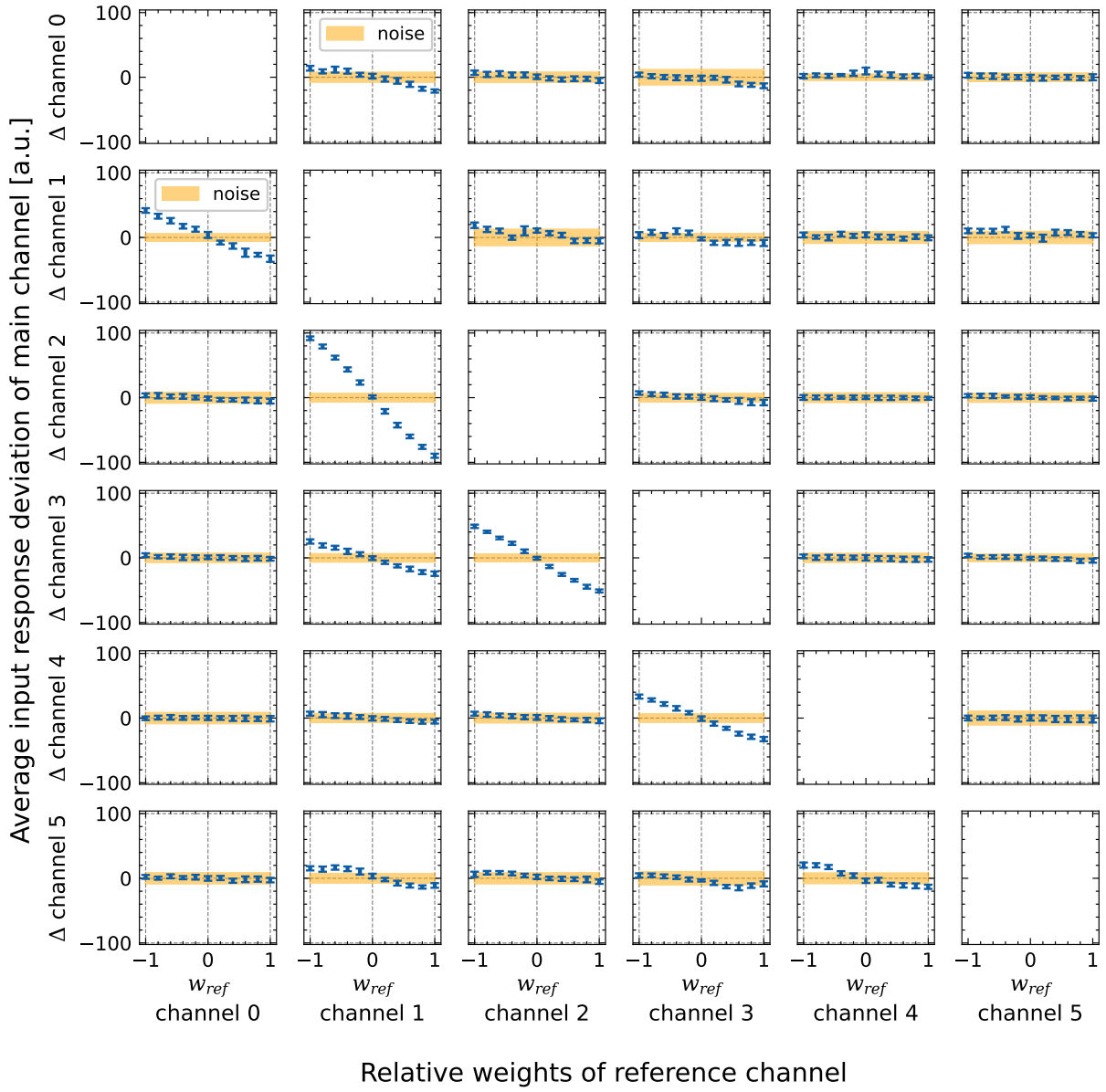


Figure 3.6.: Typical crosstalk measurement. The x-axis shows the relative weights of the respective reference channel, and the y-axis depicts the average deviation of the weight map in units of the output response for the respective main channels. The orange region indicates deviations within the weight mapping that can be explained just by the ground noise of the system. As the full output response range is about 8000 [same arbitrary unit] (see Figure 3.5), the deviation caused by crosstalk is within the range of around one percent at maximum. The majority of deviations are almost indistinguishable from noise, although a linear trend can also be observed in many of them. When a main channel c_{test} is tested, only one other reference channel is varied at a time, and all others are set to zero. The gained information enables the installation of linear corrections during the weight setting process.

weight setting and not of the input modulation. The latter case could not be accounted for without knowing all inputs, which would render such a correction scheme useless. The validity of this assumption will be investigated by assessing the impact of the linear corrections on the weight setting precision with multiple methods.

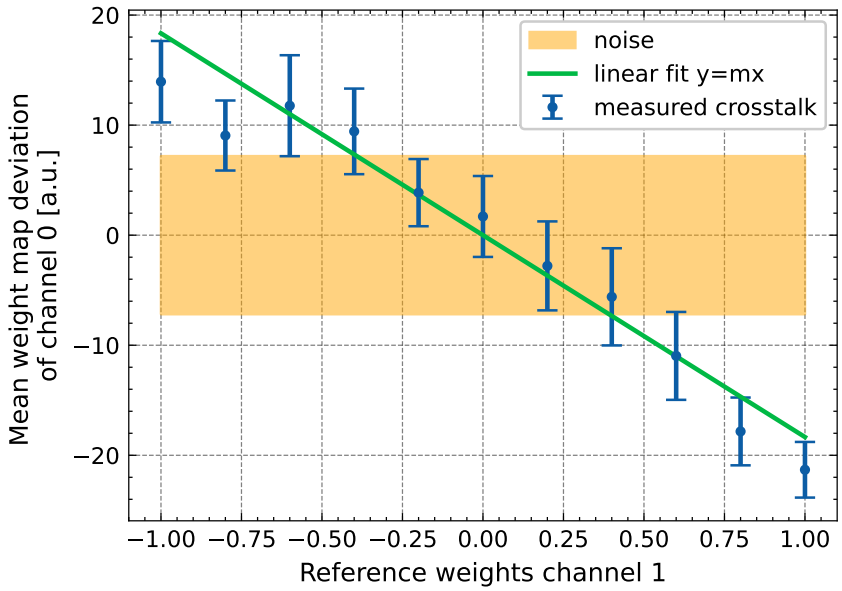


Figure 3.7.: Exemplary crosstalk from channel 1 to channel 0 and its linear fit. For each combination of main and reference channels, this fit determines a correction term for the weight setting algorithm. If the correction lies within the noise region, it is ignored.

Complete Weight Setting Scheme

Having collected information about the non-idealities of the weights and the system's crosstalk, the final weight setting algorithm is implemented. As stated before, its main goal is to conserve the relative proportions of the input weights because they will finally determine the accuracy of the MVM calculation. While focusing on the proportions, it is best to use the maximum possible range of the EAMs before clipping behaviour occurs. Based on the weight map, the algorithm can achieve that task by finding the maximum input response range that fits all channels. Within that range, the desired weights can be directly translated to target weights, and then applying the inverse function of the weight mapping yields the suitable relative weights. The correction terms obtained from the crosstalk analysis are added before the inversion takes place. For clarity, from now on, we will use the following terminology to talk about different sorts of weights:

- **Input weights, desired weights**: These are the weights that are given to the frontend interface of the photonic accelerated function and are supposed to be processed. They do not have to be within a certain range, as only their relativity matters. The necessary quantization, however, does not allow for too finely grained differences. Of course, the actual value of the weights matters in a matrix-vector multiplication, but that aspect will be dealt with separately in the output scaling process.

- **Target weights**: This is the term for the scaled version of the input weights that fit optimally within the input response range. Consequently, they are given in the same digital unit as the input response. In the current case, they lie within a typical range of thousands a.u. Possible crosstalk corrections are applied to these values.
- **Relative weights**: The digital interface of the system requires weights within the range $[-1, 1]$ that can be transmitted to the modulators; these are called relative weights. They are directly converted from the target weights through the inverse map function.

Essentially, the weight setting algorithm is a function that takes input weights as its argument and outputs relative weights matching the conditions of the PIC. The whole scheme is summarized in Figure 3.8.

The error of the truly set weights $\tilde{w}$ is estimated with an L2-metric relative to the range of the target weights w:

$$\epsilon_{\text{weight}} = \frac{||\tilde{w} - w||_2}{\Delta w} \tag{3.5}$$

$$\text{with} \quad \Delta w = \max(w) - \min(w) \tag{3.6}$$

In the case of strongly inhomogeneous channel behavior, the limited range of weak channels forces the algorithm to decrease the target weight range for all channels, undermining the system's full potential. The algorithm has a feature to disregard certain channels in the calculation of the maximum target weight range, but in order to gain an advantage from this, the broken channel (see Figure 3.5) must be assigned a value that does not exceed its range. This feature will be leveraged by the general MVM algorithm and makes the weight setting scheme an even more versatile tool when working with non-ideal inhomogeneities. A possible future iteration of this method could also sort the input weights based on the capabilities of the channels; however, this would of course require additional digital, non-photonic calculations.

3.2.2. Matrix-Vector Multiplications and 2D Convolutions

Photonic Matrix-Vector Multiplications

An original version of a general matrix-vector multiplication function was already present, which is why this algorithm is presented first. It was refined for this work, notably in the form of implementing the new weight setting algorithm and a new way of scaling the output of the PIC, which will be discussed towards the end of this section. The workings

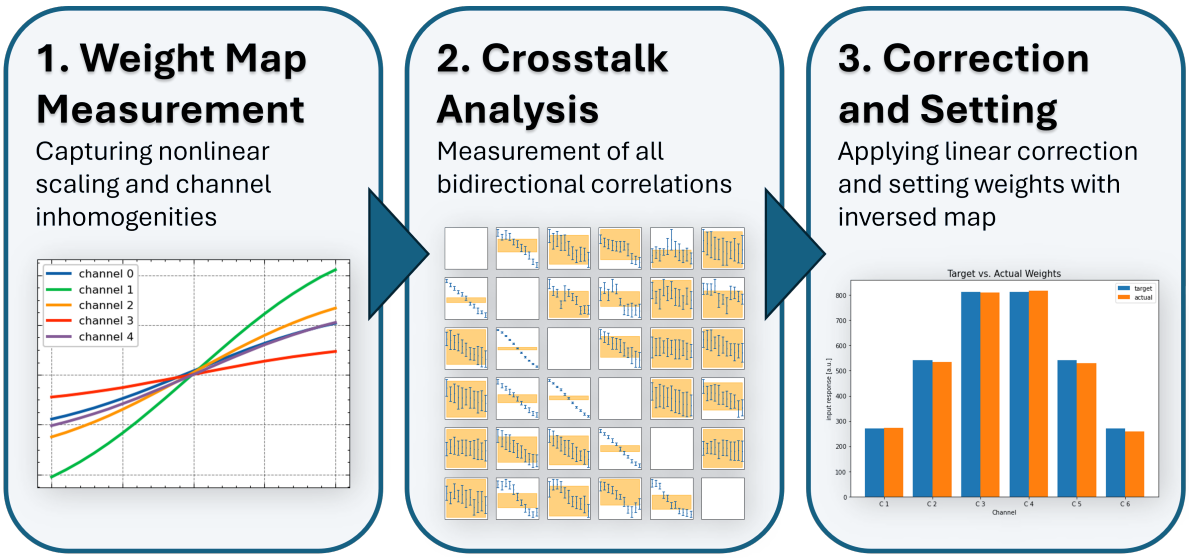


Figure 3.8.: Principles of the weight setting algorithm.

of the algorithm will be explained briefly.

The photonically processed MVM is supposed to take in arbitrary input matrices and weight vectors. While the latter can be directly given to the weight setting process, the former need to be transformed to match the requirements posed by the analog hardware first. This includes mainly normalization and quantization to eight bits (this number could potentially be higher given the hardware, but it is currently set to eight for technical reasons). An important aspect for both input and weight matrices is tiling: Since the requested size of the weight vectors is usually higher than the number of channels on the PIC - especially in the context of fully connected layers in Machine Learning applications - the larger vectors are segmented into smaller parts that just fit the crossbar array. The input matrices must be tiled accordingly, such that a matrix-vector multiplication of dimension $(m \times n) \cdot (n \times 1)$ is really processed as n/c different operations with shape $(m \times c) \cdot (c \times 1)$, when c is the number of channels. The resulting parts are of shape $(m \times 1)$ and need to be added to obtain the desired result. Tiling for the weights also happens in the second dimension; thus, also matrix-matrix calculations are possible.

As explained in 3.1.2, the underlying operation realized by the FPGA is a convolution. Therefore, the input matrices are effectively mapped onto a virtual image that the convolution kernel runs through. This means that not all photonic operations are ultimately useful; only those where the weight kernel perfectly aligns with a specific set of input values. In the case of an internal convolution kernel with nine weights, only a ninth of the actual operations are relevant. Nevertheless, working with a virtual image also gives helpful opportunities, for example, automatically giving an input of zero to broken or weak channels, removing the concern about their weights and their negative impacts during the weight setting process.

The accuracy of the photonic MVMs will be evaluated by comparing them with digitally calculated ones that are processed on the CPU of the RFSoC. Given the ideal MVM $y_k = X_k w$ and the measured output $\tilde{y}_k$, a popular way to define the total error ϵ_{MVM} of k operations is

$$\epsilon_{\text{MVM}} = \frac{\langle ||y_k - \tilde{y}_k||_2 \rangle_k}{\langle ||y_k||_2 \rangle_k} \tag{3.7}$$

[23]. This formula will test the MVM algorithm with random inputs and weights.
The above error captures all non-idealities, but it is also possible to approximate the impact of sub-errors such as the weight setting error: In contrast to the ideal result $y_k = X_k w$, the measured one can be seen as $\tilde{y}_k = X_k \tilde{w}$, neglecting the error of the input matrix [6]. By solving the regression problem

$$\tilde{w} = \text{argmin}_{\tilde{w}} ||\tilde{y} - X\tilde{w}||_2, \tag{3.8}$$

we receive another estimation of the true, actually set weights inside the crossbar array [6]. Based on this, the weight setting error ϵ_{weight} can be obtained with formula 3.5. This will be an additional guideline to quantify the performance of the new weight setting scheme, and it will also help assess the impact of the crosstalk corrections. The fact that this metric is not accounting for the noise on the input light will be reflected in an unexpectedly strong correlation between weight error and averaging.

Photonic 2D Convolutions

As part of many neural network architectures in the field of computer vision [33][18], convolutional neural networks (CNNs) play a crucial part in machine learning. Especially for the targeted task of this work, image recognition, they are superior to plain artificial neural networks. Therefore, a function that computes convolutions is generally desirable within the framework. Furthermore, we know that these operations are ideally suited for the optical hardware.

The mathematical operation of a convolution can be described by a normal matrix-vector multiplication, which is why it would be possible to use the MVM algorithm and only preprocess the input to receive the outcome of a convolution. However, the following thoughts highlight issues with that approach. For example, a 2D-convolution like in Figure 2.6 with a 3×3 kernel, no padding, and a stride of one, can be realized by sending the input pixel in a shifted manner several times to be multiplied with the static weights. The first row C_1 of the convoluted picture is then calculated in terms of matrices as

$$C_1^T = \begin{pmatrix} I_{1,1} & I_{1,2} & I_{1,3} & I_{2,1} & \dots & I_{3,2} & I_{3,3} \\ I_{1,2} & I_{1,3} & I_{1,4} & I_{2,2} & \dots & I_{3,3} & I_{3,4} \\ I_{1,3} & I_{1,4} & I_{1,5} & I_{2,3} & \dots & I_{3,4} & I_{3,5} \\ & & & \vdots & & & \\ I_{1,m-2} & I_{1,m-1} & I_{1,m} & I_{2,m-2} & \dots & I_{3,m-1} & I_{3,m} \end{pmatrix} \cdot \begin{pmatrix} w_{11} \\ w_{12} \\ \vdots \\ w_{32} \\ w_{33} \end{pmatrix},$$

while m is the width of the input image. Each input value would need to be sent k^2 times for a kernel of size $k \times k$ (excluding edge cases), and as mentioned in 3.1.2, this would produce a larger data stream, consisting primarily of duplicated values, between the memory and the programmable logic. Together with the fact that the FPGA actually performs convolutions at the ground level and the resulting necessity to transform an input matrix into a virtual image within the MVM algorithm, it is clear that routing convolutions via the matrix-vector multiplication interface is highly inefficient. That is why a native convolution scheme was created despite the already existing general MVM function.

The algorithm is specifically designed to fit into Machine Learning frameworks. Thus, it precisely replicates the behavior of 2D convolution functions in popular AI libraries, such as PyTorch, enabling the direct use of photonic computation within models of their architectures. This includes preparing an interface for multichannel batches, kernels, and biases, and adjusting the padding in a way that the internal convolution mechanism depicted in Figure 3.2 performs the desired calculation. Automatic stacking of the images of a batch allows to profit from the high input bandwidth of the photonic accelerator.
Even though much fewer weights are used than in a classical MVM operation, tiling is still required. For example, a 3×3 kernel must be split, given the test setup with six channels on the crossbar array. Special attention must be paid to channels that cannot be set to zero, as this was the case here. Without creating another virtual image like in the MVM algorithm, these channels cannot be given zero inputs. They can either be turned off completely or, as done in this case, they are automatically assigned the largest absolute desired weight and can still contribute to the computation. Tiling is also crucial for working with multichannel kernels and images.

The most important parameter for both the general MVM and the convolution algorithm is the number of measurement averages, as it is the main determinant of noise in the output signal and thus of the result accuracy. Figure 3.9 underlines the impact of that parameter and shows how the photonically computed convolution of a handwritten zero performs against a digital calculation.

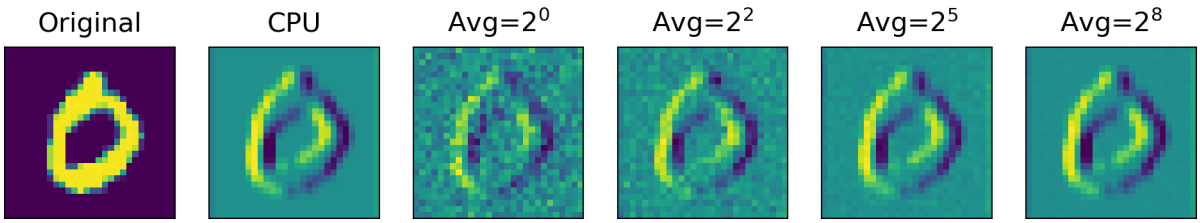


Figure 3.9.: Photonic Convolution of an MNIST image with varying averaging setting. The image is taken from the MNIST data set (see Section 3.3.2) and convoluted with a vertical edge detection Sobel G_x filter. Increasing the averaging makes the optical convolution competitive with the digital computation.

The quality of the convolutions is analyzed in a manner similar to the calculation of matrix-vector multiplication errors with formula 3.7. Here, every pixel of a convoluted image can be regarded as a photonic MVM output $\tilde{y}$.

Output Scaling

As seen before, the scale of the signals measured by the ADCs has arbitrary units that originate from the maximum input range of the converter but do not carry meaningful information about the actual scale of the computation. This becomes especially clear by recalling that the weight setting is purely based on relative proportions and that weights like $[1, 1, 1, 1, 1, 1]$ internally cause the same modulator voltages as $[10, 10, 10, 10, 10, 10]$. Therefore, a method is needed to transform the output into an absolute scale.

In principle, the multiplication with a proportionality factor is sufficient because of the linear nature of the operations and the balanced readout implemented within the FPGA. Adding a small offset can account for non-ideal behavior, though. Previous work has aimed to capture the optimal affine transformation for every set of weights by calculating a reference input matrix both photonically and digitally on the CPU, and minimizing the squared error between the target and the rescaled measured output. However, this linear regression approach generated conventional MVMs for each new weight tile, and even if the reference matrix is smaller than the actual input matrix, this still contradicts the concept of the photonic accelerator. It is also possible to obtain the scaling factor purely based on the input normalization and the weight map. The current scheme saves the transformation used to convert input weights into target weights and reapplies it to the measured output. Taking into account also the initial downsizing of the input, this method delivers a competitive estimate of the optimal scaling without the need for digital matrix-vector multiplications. Unlike in the previous method, no offset value can be inferred, but the physical model of the photonic system does not inherently imply such an offset. The performance of the new and the previous algorithms will be compared to evaluate the relevance of that fact.

3.3. Photonic Machine Learning

3.3.1. Hardware-Aware Machine Learning Simulations

A way to account for analog hardware non-idealities, such as noise, and for constraints like quantization during machine learning inference is hardware-aware training: It is performed digitally, but simulates analog hardware characteristics during the training of artificial neural networks, aiming to improve their robustness for deployments on real analog platforms. A popular software library for this purpose is AIHWKIT-Lightning, developed by Büchel et al. at IBM Research, which combines a wide range of simulated non-idealities with simple integration into the PyTorch framework [7]. This work will utilize this library to train adapted models to enhance the machine learning performance of the photonic tensor core.

The training with AIHWKIT-Lightning is conducted using the following methodology: First, the desired analog characteristics are saved as parameters in a configuration file. Then, a conventionally defined and potentially pretrained PyTorch model is passed to the software toolkit, which transforms it into an analog model simulator ("analog model") based on the specified configurations. In Section 4.3, the impact of this conversion on the inference accuracy is analyzed. The core of the analog models consists of analog layers, which perform the same functions as the corresponding layers in the original PyTorch model, but with hardware constraints and non-idealities applied. These layers, for instance, include linear or convolutional layers [7]. The training is performed using an adjusted optimizer that can be based on any standard PyTorch optimizer.

The most relevant features of the software kit for this study are

A. the simulation of input and weight quantization, and

B. the injection of uncorrelated noise onto the outputs and weights.

A: Within the integrated photonic system, digital 32-bit floating-point numbers are converted to 8-bit analog signals by the DACs, significantly decreasing the numerical resolution and dynamic range available for both input data and model weights. Incorporating this quantization during the training process and applying it to inputs and weights ensures that parameter differences do not fall below the resolution limit of the 8-bit representation and that the model becomes sufficiently robust to reduced-precision input. Since the experimental system has a fixed quantization, this configuration was kept constant across all training processes.

B: As discussed earlier in this chapter, several sources of noise affect the operation of the optical setup: Most notably, the ASE source and the subsequent amplifier introduce power fluctuations, while components such as the photodiodes or the TIAs also contribute to a certain level of randomness, although not as significantly. Moreover, the weight programming is not perfectly precise. To account for these effects, AIHWKIT-Lightning offers the option to inject sweeping Gaussian noise into the outputs and weights. This noise is scaled relative to the maximum value in the weight matrix, with an adjustable scalar allowing for control over the amount of noise applied. The output noise observed in the physical system is highly dependent on the averaging setting, which is why analog models with different output noise levels were trained. In contrast, the weight noise is a rather static property of the system and was not varied after an initial definition. The estimate for this initial value was obtained through the separate evaluations of the weight setting error of the newly implemented algorithm.

Other features of the software were used in their default configurations and not further analyzed. This includes, for example, a weight-clipping setting that ensures a tight weight distribution, generally favorable for analog computations. Further details about AIHWKIT-Lightning and the functioning of all its hyperparameters can be found in [7].

3.3.2. Datasets, Models and Techniques

The ultimate goal of this study is to deploy a machine learning model, specifically a convolutional neural network, on the photonic tensor core and perform classical computer vision tasks, such as image recognition and classification. Two popular benchmarks have been investigated: the MNIST dataset, containing 28×28 pixel grayscale images of handwritten digits, and the CIFAR-10 dataset, which consists of 32×32 pixel RGB images of ten categories, including planes, cats, or ships. Figure 3.10 displays some exemplary images from both datasets, along with their corresponding labels. The two sets contain 70,000 and 60,000 total images, respectively, but both have 10,000 images reserved for testing purposes. In all machine learning runs on the photonic hardware, inference was only tested on these dedicated test sets, while these pictures were specifically excluded from the training process. Following best practices, the input data was standardized before its utilization.

The main experiments for this study were conducted using a small convolutional neural network, referred to as *conv01*, which consisted of two convolutional layers with respective filter depths of 16 and 32, and one fully connected layer with ten neurons to output the final classification. As CIFAR-10 is known to be a much more complex task than MNIST, a slightly deeper CNN (*conv02*) was created for a few experiments that aimed to achieve

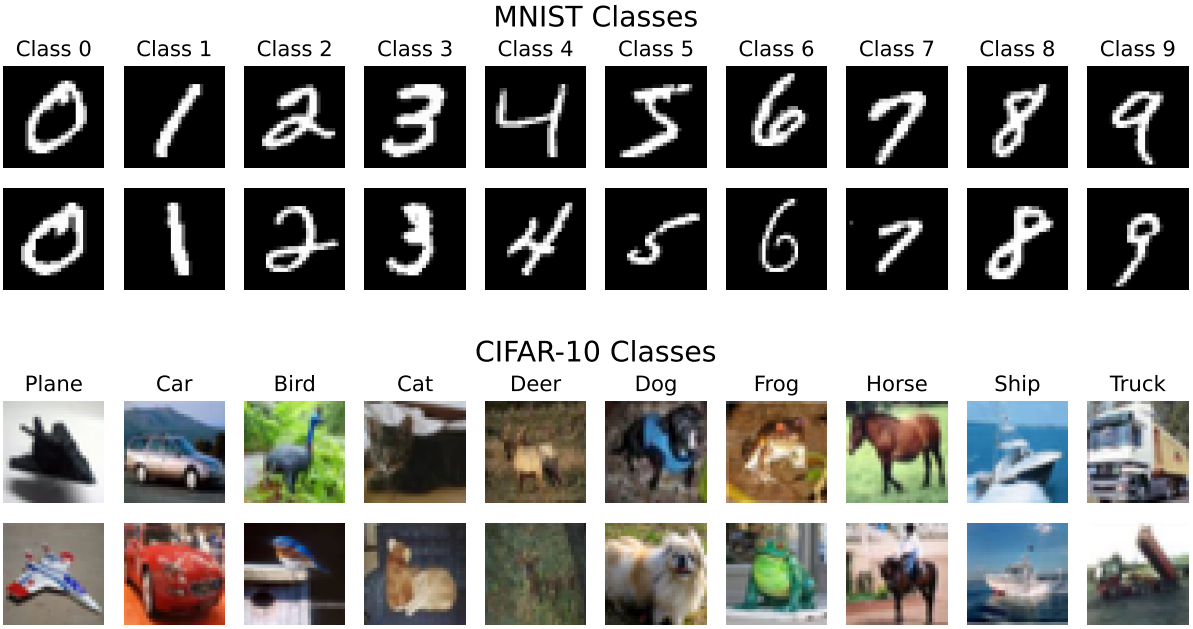


Figure 3.10.: Exemplary images from the used benchmark datasets. The upper two rows include images from the MNIST dataset with handwritten digits, the lower two rows depict images from CIFAR-10 and their respective labels.

higher scores on it. This second model consisted of four convolutional layers with kernel depths of 32,64,64,128. The details about the two analyzed models are summarized in Table 3.1. While the small model was a very simple and plain CNN that was supposed to give a baseline, the second model incorporated some advanced techniques such as batch normalization and dropout, with the former being applied in its default PyTorch configuration and the latter with a standard value of 20%.

Models were first pre-trained digitally, then converted into analog models using AIHWKIT-Lightning, and subsequently fine-tuned to account for the analog hardware characteristics. Initial experiments have shown that the stochastic gradient descent optimizer (SGD) combined with momentum yields the best results for the intensive pre-training, while ADAM performs better in the fine-tuning process. Cross-entropy was chosen to serve as the loss function. Based on the noise analysis made on the MVM and the convolution algorithm, three distinct output noise levels were chosen to train the analog models on.
For the training of the second CNN, data augmentation was used to maximize the generalization of the neural network, given its still comparatively small architecture. This included a random crop after uniform padding with four pixels, and a random horizontal flip half of the time.

After the training of a hardware-aware model had been completed, the weights and biases of its best-performing version were extracted and reloaded into a digital model, which could then be executed again within the conventional PyTorch framework installed on the

digital chip of the photonic integrated system. To enable optical processing, the presented photonic algorithms were implemented during the model definition, with the averaging setting serving as a configurable parameter. The MVM algorithm thereby adopted the general matrix-matrix calculations occurring in FC layers, while the convolution function replaced the respective internal PyTorch implementation. Activation functions, pooling layers, and the addition of biases were handled in the digital domain.
To test the impact of partly photonic inference runs and to increase experimental flexibility, all layers were programmed to optionally perform their calculation digitally on the CPU of the RFSoC. Final tests included repeated inference runs from the shallow CNN on subsets of both benchmark sets under varying averaging settings. In each case, the accuracy of the classification as well as the processing time were recorded. For the deeper CNN, selected experiments with low and medium averaging settings were conducted, achieving the highest observed scores on CIFAR-10 and enabling an assessment of the noise accumulation in multiple sequential photonically processed layers. The latter study provides an outlook of what this photonic tensor core is capable of, even with its current limitations in size and processing speed.

Table 3.1.: CNN model specifications. For the convolution layers (Conv), the number within the brackets indicates the depth of the filters, while fully connected layers (FC) are labeled with their number of neurons. All convolutions use kernels of size 3×3, same padding, and a stride of 1. The max pooling layers (Pool) use a 2×2 filter and a stride of 2. All convolutional layers were followed by a ReLU activation function.

Model	# Conv. layers	Structure	# Parameters (MNIST)	# Parameters (CIFAR-10)
conv01	2	Conv(16), Pool(), Conv(32), FC(10)	20490	25578
conv02	4	Conv(32), Conv(64), Pool(), Conv(64), Conv(128), Pool(), FC(10)	-	212682

4. Results

This chapter will first present the preliminary analysis of the system stability and then focus on the evaluation of the new control algorithms in Section 4.2. Afterwards, the hardware-aware training process is discussed in Section 4.3, followed by the results from deploying machine learning models on the photonic tensor core in 4.4.

4.1. Weight Map Stability

The basis for all experiments on the photonic system is the new weight setting scheme that relies on the weight map, which stores information about the channel inhomogeneities and nonlinear scaling effects. To assess the temporal stability of that map, a preliminary test was conducted over two hours to measure the occurring deviations. For this purpose, a new weight map was measured for every minute and compared with the initial map with an L2 error. The result is presented in Figure 4.1. The plot shows that there is an error of about 1% that occurs almost immediately after the start of the experiment, which indicates a ground level of irreducible noise. Over time, mean noise levels around 2% are observed, but no significant drift of the weights is exhibited. This implies that an hourly updating of the weight map is sufficient regarding machine learning tests with long run durations.

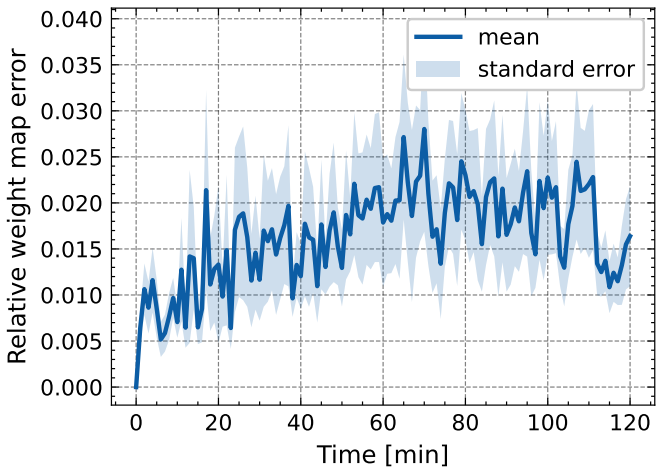


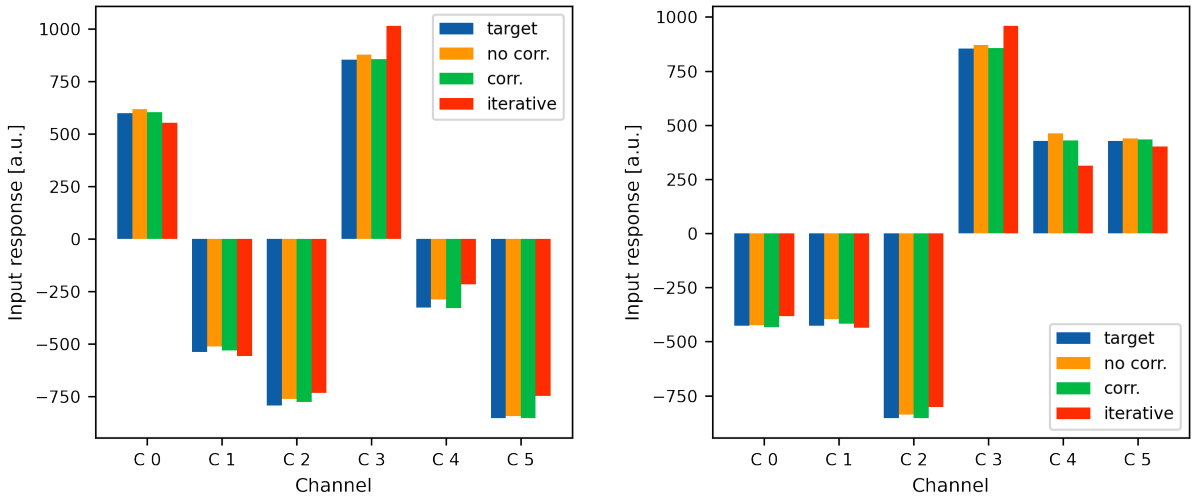
Figure 4.1.: Temporal weight map stability. For every minute, a new weight map was measured and compared channel-wise to the initial map with an L2-error. The plot depicts the mean error from all channels together with its standard error.

4.2. Performance of the Control Algorithms

4.2.1. Weight-Setting

The introduced weight-setting algorithm is based on the weight maps and addresses non-linear weight scaling; furthermore, the discussed linear crosstalk correction can be applied in this scheme to achieve even higher accuracies. The new algorithm was tested in two versions, once with and once without the crosstalk correction. As a reference, the old iterative algorithm was also tested with the number of iterations set to 60. To examine the quality of these three methods, the actual weights were measured as input responses after each weight-setting with the respective function from the software interface (see equation 3.4); and then compared with the target responses. An exemplary comparison of the target weight matrix and the actual weights can be found in Figure 4.2. The process of setting weights was repeated 100 times with uniformly distributed weights between -1 and 1, and formula 3.5 is used to evaluate the error. The resulting mean error and the mean time per iteration are reported in table 4.1.

If weaker channels, that exhibit an extraordinarily smaller input response range than the others, were taken out of consideration, the weight setting accuracy strongly improved, which is shown in Table 4.2. This benchmark was carried out similarly to the previous one, with the only alteration being the exclusion of the weakest channel. Although chan-



(a) Random weights **(b)** Sobel weights

Figure 4.2.: Exemplary comparison of different weight-setting algorithms. The blue targets show the desired input response for each weight channel, and the red bars represent the weights set by the original, iterative algorithm. The green and orange bars show the results of the new method, respectively with and without the crosstalk correction. In (a), random weights between -1 and 1 were set. In (b), the programmed weights were $[-1, -1, -2, 2, 1, 1]$, which are, for example, commonly used inside a Sobel filter for edge detection [27].

Table 4.1.: Errors and time consumption of weight setting process. The previous iterative method was compared to the new ones based on the weight mapping for 100 random weights between -1 and 1. The true weights were estimated by the specific measurement function described in 3.2.1. The term correction refers to the linear corrections obtained from the crosstalk analysis.

Method	**Error**	**Time [s]**
Previous, iterative algorithm	$20.0\% \pm 1.4\%$	7.680 ± 0.009
Weight map algorithm, without correction	$3.34\% \pm 0.10\%$	0.0631 ± 0.0008
Weight map algorithm, with correction	$1.66\% \pm 0.06\%$	0.0631 ± 0.0006

Table 4.2.: Errors and time consumption of weight setting process for modified weights. This time, only the five strongest channels were taken into account for the same benchmark test as in table 4.1.

Method	**Error**	**Time [s]**
Previous, iterative algorithm	$6.3\% \pm 0.5\%$	7.667 ± 0.010
Weight map algorithm, without correction	$1.57\% \pm 0.06\%$	0.0631 ± 0.0005
Weight map algorithm, with correction	$0.407\% \pm 0.015\%$	0.0630 ± 0.0005

nel inhomogeneities will not disappear completely in analog computing, we can expect future iterations of the prototype PIC to exhibit more homogeneous behavior, for example, due to design and fabrication enhancements. Therefore, these results provide insight into what a further developed integrated system will be capable of.

It could be observed that the new algorithm significantly increased the accuracy of the set weights compared to the old algorithm and was also much more stable and consistent. Additionally, the linear correction proved to be very effective, reducing the error by approximately half. However, the actual scale of the accuracies should not be overestimated, because it is specific to the way of measuring the weights (see 3.2). A different evaluation method based on general MVMs presented in the following section suggests that the errors of the setting process are larger, the relative performance advantage of the new algorithm prevails, though.
An even bigger improvement was achieved in reducing the time required to set the weights. The new algorithm takes only about 63 ms, which is more than two orders of magnitude less compared to the iterative scheme. It should be noted that the weight map and the crosstalk calibration both induce an overhead operation of about 15 seconds and 17 minutes respectively, but this can be neglected as both calculations do not need to be done often: On one hand, we have seen in Section 4.1 that the weight map is quite stable, thus it only needs to be updated on an hourly basis. On the other hand, the crosstalk analysis

must be performed only once per setup, so it also does not contribute to the actual time consumption. Even with the additional preprocessing in mind, the new method is still significantly faster than the old one.

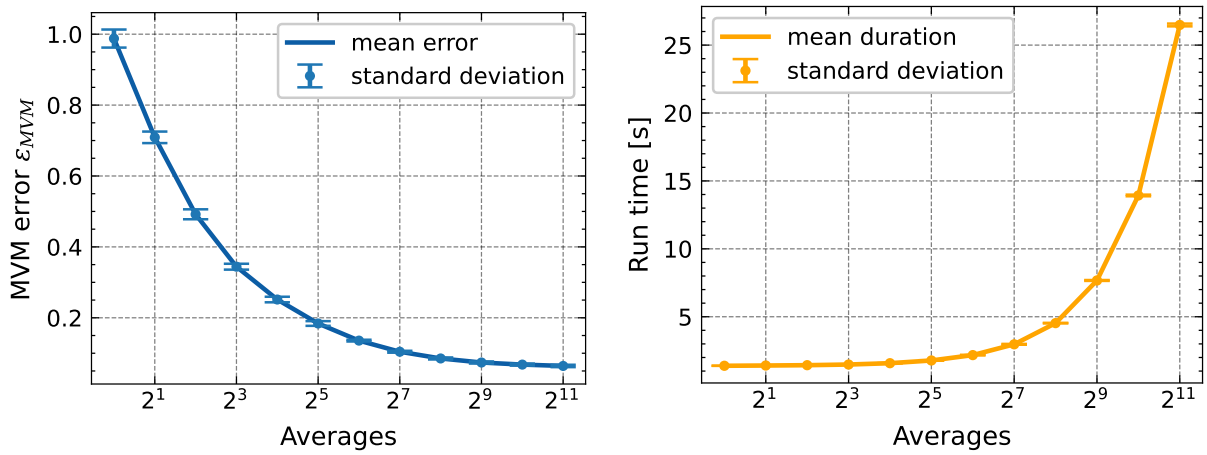
While the increased accuracy is surely a great benefit, the real highlight lies in the time savings, as this is the key element that makes machine learning operations possible on that chip in the first place. Even very small neural nets have at least thousands of weight kernels; thus, calculations would hardly be feasible in a reasonable amount of time when waiting several seconds just to set six weights was required. Therefore, this new, more accurate, reliable, and foremost fast algorithm has become the heart of the further experiments.

4.2.2. General Matrix-Vector Multiplications

Before machine learning tasks could be addressed, the accuracy of the matrix calculation algorithm and its relationship to the averaging parameter had to be evaluated. Figure 4.3 shows the result of ten test runs, where a random input matrix of size (1000×10) was multiplied by a random weight matrix of size (10×10); their values came from a uniform distribution $\mathrm{Unif}[-1, 1]$. The measurement was repeated for every common averaging setting and then compared to a digitally calculated result. Figure 4.3a depicts the MVM error according to equation 3.7: It can be seen that plain runs with no averaging lead to very high errors of up to 100%, but for averaging settings of 5 or more, implying $2^5 = 32$ averages, the error drops below 20% (for a detail view of the graph, see supplementary Figure A.1). As later experiments showed, a 10% error is a suitable goal when working with machine learning. This value was achieved by $2^8 = 256$ measurements per input matrix. Figure 4.3b makes clear that the higher accuracies come at the price of calculation speed. The time scales linearly with the number of averages, which can also be observed in Figure A.2. It is notable, however, that the run times would be significantly longer without the new weight setting scheme. The two plots provide valuable insight that averaging settings between 5 and 8 offer comparably accurate calculations while still being time-efficient. Higher repetition rates only make sense if the highest accuracy possible is required.

Output Scaling Analysis

Besides incorporating the new weight setting scheme into the MVM algorithm, the new output scaling method, which estimates the scaling factor based on the weight map, was also adapted. The original scheme determined an affine transformation after comparing a small, photonically calculated test matrix to a digitally calculated one. Figure 4.4 plots



(a) MVM error of large matrix calculations

(b) Time consumption per test

Figure 4.3.: Performance of the MVM algorithm. The data was obtained during ten test runs with random MVM calculations of shape $(1000 \times 10) \cdot (10 \times 10)$. The accuracy of the photonic computations lays out the conditions for machine learning applications. Note the exponential scaling of the x-axis according to the possible values for the averaging parameter. This is to better assess which parameter settings are most suitable for subsequent experiments. For an alternative depiction of the time consumption, compare Figure A.2.

the MVM error of both methods, together with an optimal scaling error. The test procedure was the same as for the initial MVM error evaluation. For low averaging settings, the newly introduced scaling performed slightly worse, but it surpassed the previous algorithm notably when stronger averaging was applied, making it preferable for subsequent use cases. The analysis of the optimal scaling curve reveals that the photonic calculation is, in theory, better than assumed for small averaging values, up to approximately 30% for avg = 1. In practice, this was not achieved with the estimated scaling parameters. In this region, there was a bigger offset in the perfect affine transformation, which is generally not captured when determining the scaling from the weight map. However, these offsets were mostly a statistical effect and disappeared when the mean of several measurements was taken; this is why the new scaling method closed its accuracy gap compared to the perfect scaling. From this, we conclude that the photonic tensor core should be used with at least medium averaging settings to fully exploit its potential.

Reevaluation of the Weight Error

With the MVM algorithm in place, it is possible to reevaluate the weight error using formula 3.8 to obtain a more realistic assessment. To recall, this involves calculating matrix-vector multiplications and estimating the actual weights through regression optimization between the measured and calculated outcomes. For this test, a random input matrix of size 1000×20 was multiplied by a random weight vector of size 20×1, and this

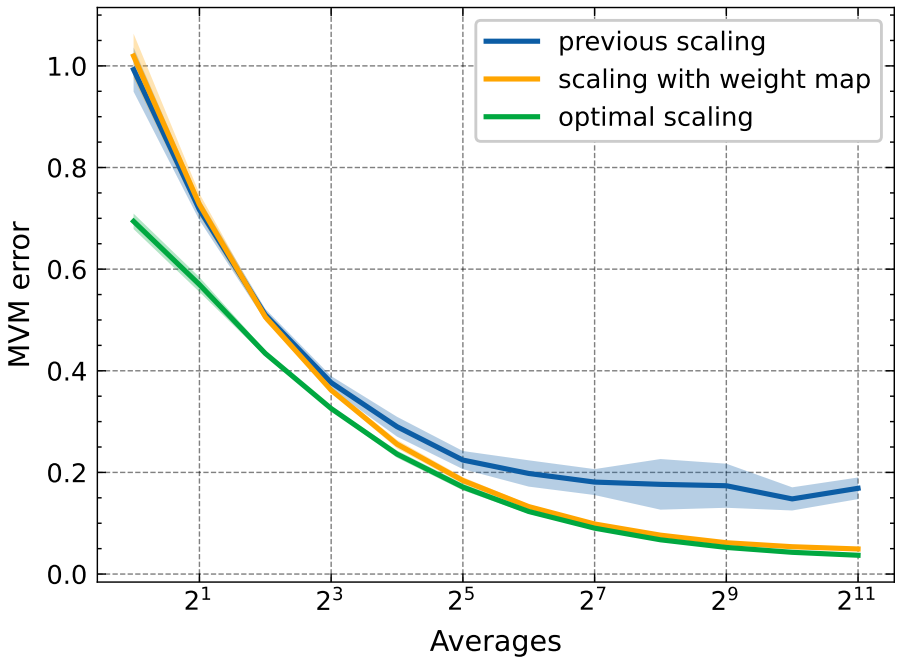
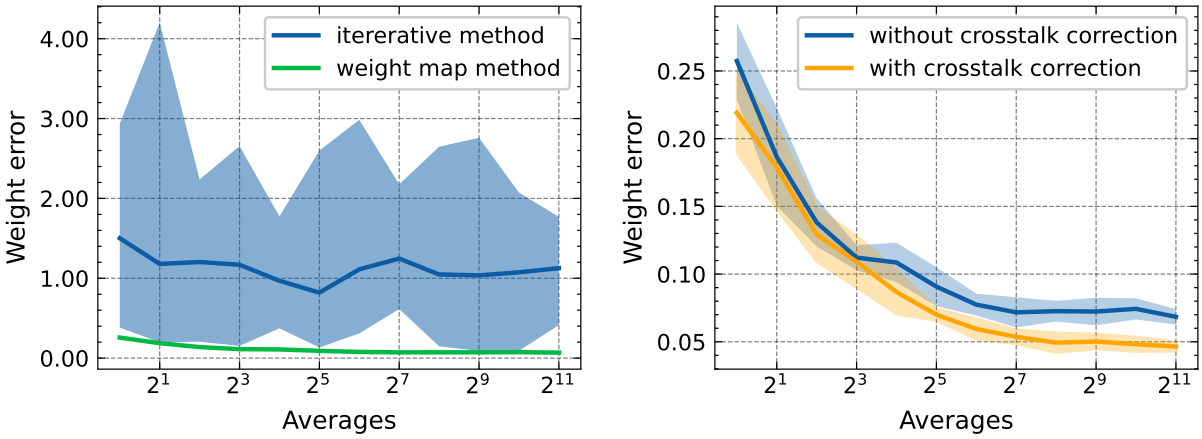


Figure 4.4.: Output scaling analysis. The previous method, determining the output scaling based on a probe matrix, is compared to the scaling deduced from the weight maps. An optimal scaling curve provides a reference for the potential of the photonic computations.

process was repeated ten times. We expect the weight setting error to be independent of the averaging, since the weights are solely controlled by the EAMs and should not be more precise just because they are measured more frequently. However, since this error metric relies on large MVM operations and not the specifically tailored function from the software interface, the ASE noise will be reflected. Therefore, the just described test is, just as before, carried out separately for all common averaging values. Figure 4.5 shows the results of this benchmark. This is another opportunity to compare the iterative weight setting with the weight map-based scheme, and furthermore, the effect of the crosstalk corrections is also analyzed. While the plots show larger weight errors compared to Section 4.2.1, they still underline the improved functionality of the introduced weight setting method (Figure 4.5a) and the usefulness of the linear correction (Figure 4.5b). A strong dependence on the averaging setting in all cases can be seen too, which is an unwanted but method-inherent effect. When focusing only on higher averages, the error curves reach a plateau, suggesting that the noise impact is negligible at that point and that this is a good estimate of the actual weight setting error.

Table 4.3 shows average weight errors and their uncertainty when only runs with a total MVM noise of less than 20% are taken into account (averaging ≥ 5). As already seen in the plot, the crosstalk correction does have a positive impact on the overall accuracy, but it only decreases the error by about 30%, not by almost 50% as implied by Table 4.1. This suggests that not all crosstalk is an effect of the weight setting, but rather also from



(a) Effect of the weight map scheme

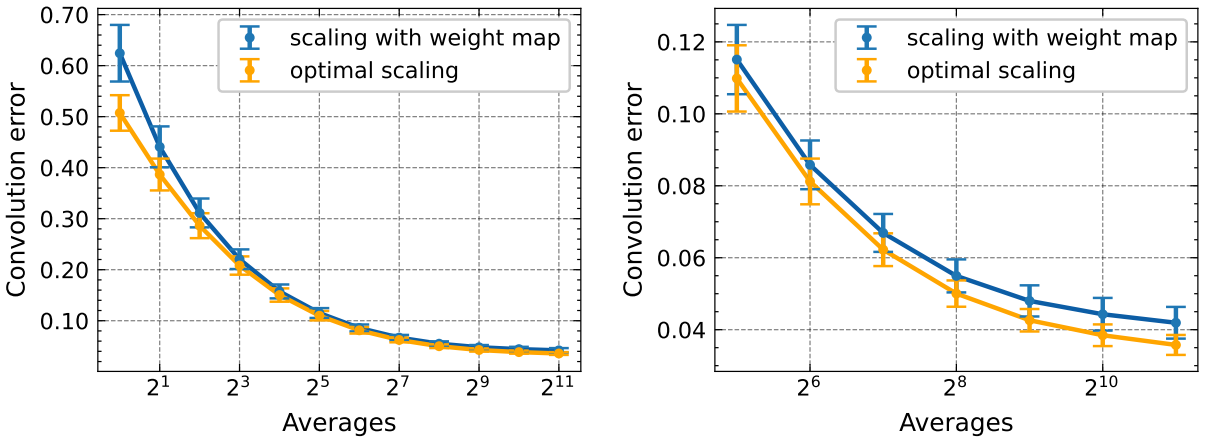
(b) Impact of the crosstalk correction

Figure 4.5.: Reevaluation of the weight error ϵ_{weight} based on large MVM calculations. Based on multiple large MVM operations, the actual weights have been estimated and compared to the input weights. Per average value, a random MVM of dimension $(1000 \times 20) \cdot (20 \times 1)$ has been calculated ten times. Part a) confirms the improvement of the weight map approach over the iterative one, both in stability and overall accuracy, and part b) highlights performance improvements based on the linear crosstalk corrections. The errors are higher than calculated in 4.2.1, but show a similar relative comparison between different methods. This estimate of the true weights is influenced by the stochastic nature of the incoming light and the electronic noise, which is why the average setting plays a role. The fluctuations of the iterative algorithm are due to strong inhomogeneous channel behavior; its performance can be tremendously improved by preselecting benign weights. For further details, see supplementary Figure A.3.

input modulation, as briefly discussed in 3.2.1. This is not captured by the native weight measuring function because it tests the PIC responses each time with the same inputs, which is why the results in this section are more applicable. Still, the crosstalk correction element of the weight setting algorithm provides a valuable performance increase.

Table 4.3.: Errors of weight setting process based on a different evaluation method. These averages are obtained from the data plotted in 4.5, but only for cases where the total MVM error is less than 20%. In this region, the dependence on the averaging setting is less pronounced, which is why a more realistic estimate for the systematic deviation of the weight setting may be found there.

Method	**Error**
Weight map algorithm, without correction	$7.54\% \pm 0.17\%$
Weight map algorithm, with correction	$5.39\% \pm 0.14\%$



(a) Convolution error **(b)** Detail view

Figure 4.6.: Performance of the convolution algorithm. The test was conducted with 100 input tensors of shape [100, 1, 28, 28], each convoluted with a 3×3 kernel, which simulates a convolution run of MNIST images. An optimal error was calculated from the photonic output as a reference. While a) shows the full range of tested averaging settings, b) gives a detailed view for avg $\geq 2^5$.

4.2.3. Convolutions

Similarly to the previous section, the convolution algorithm was evaluated for its accuracy under various averaging settings. Instead of 2D matrices, the input consisted of multidimensional tensors that represented image data in machine learning frameworks like PyTorch. For each averaging setting, a random tensor of shape [100, 1, 28, 28] was convoluted by a random 3×3 kernel, simulating the processing of 100 MNIST images. The experiment was repeated 100 times. Figure 4.6 shows the results: The error, calculated with formula 3.7 as well, follows the same tendency as the MVM error curve, but its magnitude is lower, indicating a better performance. This is due to a more elaborate distribution of weights to available channels, as described in 3.2.2, which originated from the necessity of working with channels that can only be assigned non-zero weights. The detail view in 4.6b demonstrates that for settings of avg $= 2^5 = 32$, we can already expect errors of 12% and less. Additionally, an error curve with optimal scaling is provided as a reference, underscoring the effectiveness of the weight map-based output scaling scheme once again, particularly for high averaging.

As also implemented in influential other studies, the probe inputs were uniformly distributed around zero (see, for example, [23],[6]). But the distributions of both inputs and weights can have a great influence on the accuracy. This is not within the scope of this work, but the supplementary Figure A.4 provides first insights on that matter, which could be addressed in future projects.

Although processing 7.84 times as many input values as the MVM algorithm in its respec-

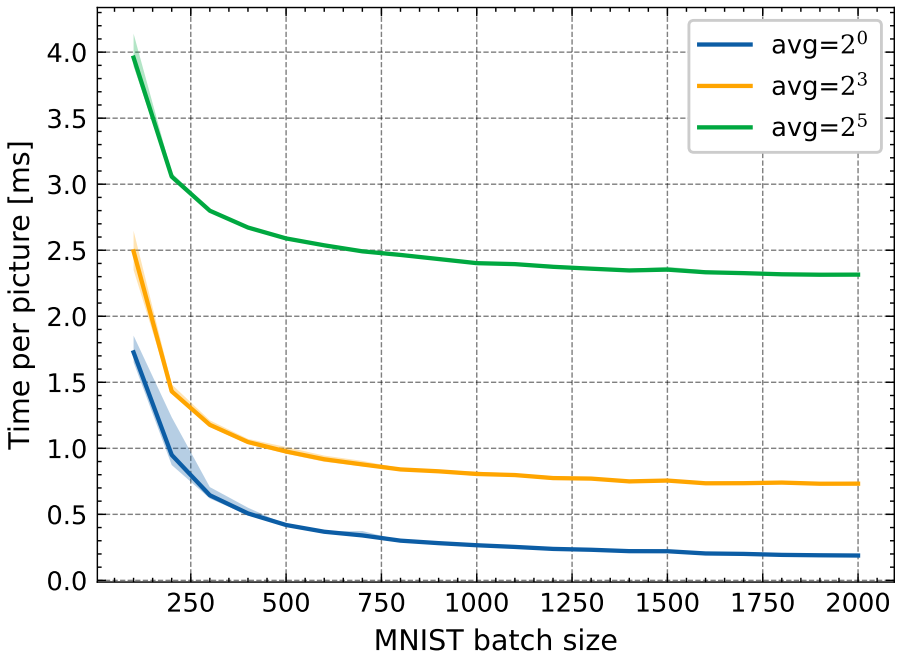
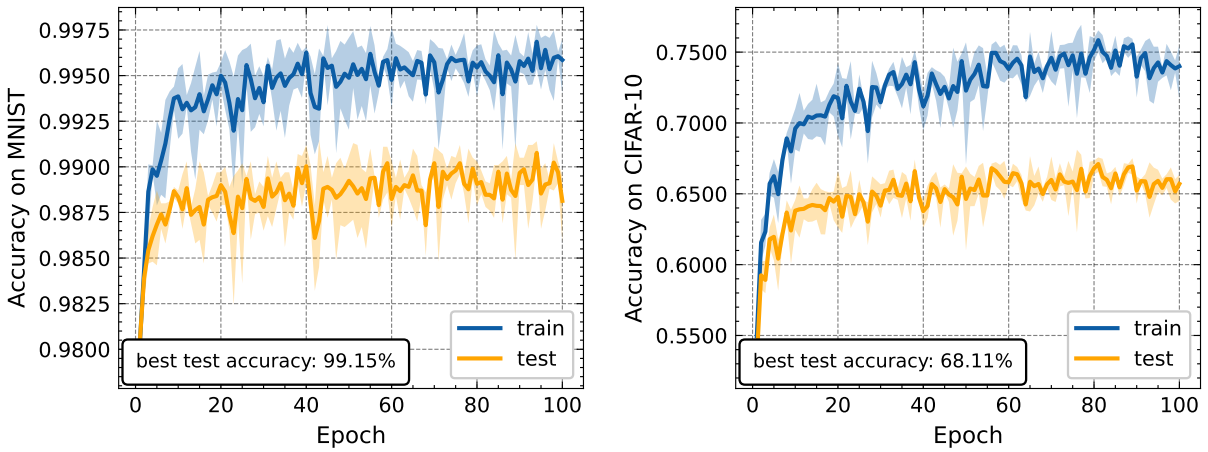


Figure 4.7.: Batch size efficiency for convolutions. The durations of convolutions of a batch containing MNIST images with a 3×3 kernel were measured for varying batch sizes and then divided by the number of images per batch (note that this kernel size implies already twofold tiling of the weight matrix). This computation was done for three distinct averaging settings. The time efficiency increases drastically when scaling from 100 images per batch to several hundred, which is due to the decreasing significance of programming and weight setting overhead calculations for larger inputs. While the averaging setting naturally influences the magnitude of the curve, it does not change its shape. Based on this, subsequent experiments were conducted with a batch size of 500 by default.

tive test, the convolution function was on average 45% faster per experiment run. This is expected because significantly fewer weight switchings were required, but it highlights again the suitability of convolutional computations for photonic hardware accelerators.
In foresight of the neural network deployment, the time consumption per image was analyzed over varying batch sizes, which is depicted in Figure 4.7. This graph illustrates that the tensor core's potential is best utilized with large input data streams; specifically, for the case of MNIST images, the best time efficiencies were achieved for batches of size 500 or larger. Hence, in photonic machine learning experiments, 500 was the standard batch size.

4.3. Hardware-Aware Simulations

For both benchmark datasets, the introduced small CNN (conv01) was first pre-trained for 100 epochs within the PyTorch framework. In each epoch, the model was evaluated on the train and test datasets. To find the best model for further finetuning, the model



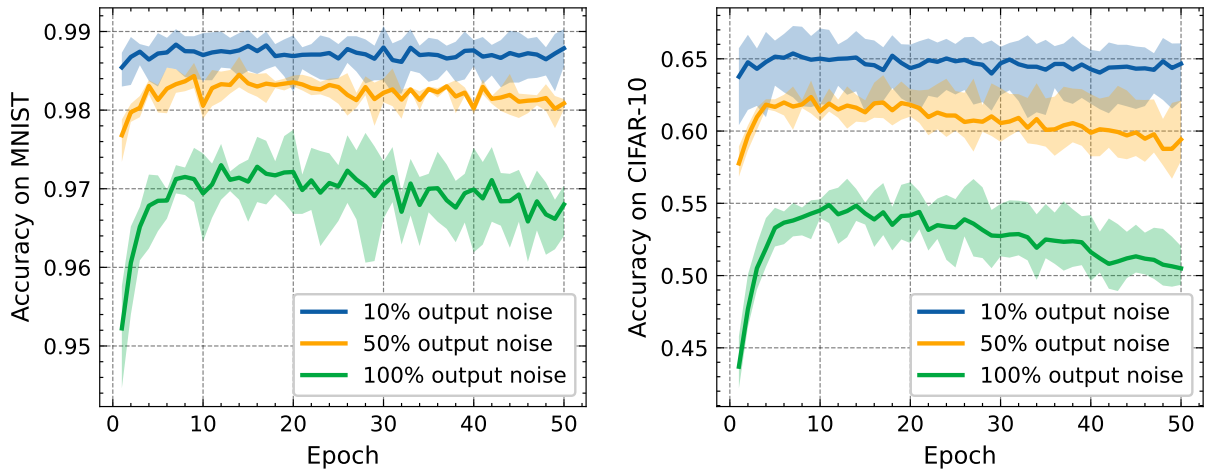
(a) MNIST accuracy

(b) CIFAR-10 accuracy

Figure 4.8.: Pre-training of the initial CNN. The plots show the train and test accuracies of the digital convolutional neural network conv01 for MNIST (a) and CIFAR-10 (b) as well as their respective maximum test performance. The training from a random initialization was performed three times each; their mean is represented by the solid line and the range between minimum and maximum values is shown by the colored shading. The accuracy of the initialized model is in both cases approximately 10%, as expected, but not included in the graph to resolve performance details of higher epochs.

scoring best on the test set was chosen from three total training runs. Figure 4.8 illustrates the mean measured accuracies per epoch as well as the minimum and maximum accuracies encountered during the separate training procedures, both for MNIST (Figure 4.8a) and CIFAR-10 (Figure 4.8a). The former, being a dataset known for its simplicity, allowed the digital model to achieve maximum accuracies of 99% and more already within the first 20 iterations, with the overall best performance of 99.15%. In contrast, the images of CIFAR-10 are much more complex than the handwritten digits, which is reflected in the models' ability to classify them: While the overall test accuracy does not exceed 68.11%, the learning curve is also increasing more slowly. Furthermore, the gap between train and test performance strongly suggests overfitting, which could also not be reduced with stronger regularization. Advanced training methods, some of which applied in the assessment of the deeper convolutional neural networks, may improve the outcome; however, the model consists of only approximately 26,000 parameters, which is likely not expressive enough to score significantly better. As for the small CNN, its primary goal was to showcase what basic methods can already yield. A focus on improving the digital performance by a thorough hyperparameter search is a possible objective for further studies.

After the completion of the pre-training, the digital neural networks were converted into hardware-aware models with custom noise settings. Based on the results obtained during the reevaluation of the weight setting algorithm, the weight noise for all hardware-aware configurations was set to 10%, which includes a buffer margin and thus ensures that the



(a) MNIST accuracy **(b)** CIFAR-10 accuracy

Figure 4.9.: Hardware-aware fine-tuning. The best digital CNN (conv01) was transformed into analog models with different output noise levels and then trained again, which was done three times in total. The solid line and the shading indicate the mean accuracy and the whole range of accuracies, respectively. The parameters which lead to the best test performance were extracted for the experiments on the photonic tensor core.

weight programming error is not underestimated during the simulations. For the output noise, the MVM error range was taken as a reference, which spans from approximately 5% to 100% (see Figure 4.3). Three distinct noise levels within this range were chosen to serve as the experimental basis for inference tests on the photonic hardware: 100%, 50% and 10%. Immediately after the conversion, models exhibit decreased performance compared to the pre-trained digital neural network, depending on their noise setting, indicating that the original weights are not well-suited for deployment on non-ideal analog hardware. Fine-tuning with consideration of the analog characteristics increases the accuracy again, even though the maximum digital level cannot be reached anymore (for further details, see supplementary Figure A.5). Here, the fine-tuning included 50 additional training epochs, which were repeated three times to select the weights that yielded the best classification results. Figure 4.9 illustrates how this process compares between the different noise levels: Clearly observable is the decrease in overall accuracy with increasing amount of output noise, which is sensible because stronger noise robustness comes at the price of less finely adjusted weights and thus lower peak accuracy. However, it is still remarkable that neural networks with noise levels as high as 100% can still achieve scores of more than 97% on MNIST and 55% on CIFAR-10. This is especially emphasized when looking at the model performance after the analog transformation, yet before the fine-tuning, which is reported in table 4.4

Similarly, the deeper convolutional neural network was also processed. Since it has more than eight times as many parameters as the smaller CNN in the case of CIFAR-10, it is much more expressive and can generalize better. Its maximum digital score reaches

89.99%, its corresponding accuracies after the conversion before and after the pre-training can be found in Table 4.5. Especially promising is the model with the lowest noise level, as its simulation suggests scores of more than 86%. The best fine-tuned parameters from both architectures for all investigated noise levels were extracted for the deployment on the integrated photonic system.

Table 4.4.: Analog conv01 simulation performances before and after the hardware-aware fine-tuning. For each dataset, the left column contains the mean accuracy of the model immediately after the conversion from the digital predecessor and its standard error, resulting from three training processes. The right column displays the maximum measured accuracy during any fine-tuning run. The model corresponding to this performance was selected for deployment on the photonic hardware. The digital accuracy, on which the fine-tunde models are based, is given as a reference.

	MNIST Accuracy [%]		CIFAR-10 Accuracy [%]	
Noise level	before fine-tuning	after fine-tuning	before fine-tuning	after fine-tuning
10%	97.11 ± 0.04	99.07	57 ± 3	67.27
50%	85 ± 3	98.68	29.1 ± 1.2	63.87
100%	45 ± 3	97.75	17.6 ± 0.4	56.70
Digital	99.15	-	68.11	-

Table 4.5.: Analog conv02 simulation performances before and after the hardware-aware fine-tuning on CIFAR-10. As before, the analog models constructed with different noise levels are compared to the digital, noise-free neural network.

	Simulation Accuracy [%]	
Noise level	before fine-tuning	after fine-tuning
10%	63.6 ± 2.9	86.10
50%	39.9 ± 3.6	80.60
100%	24.6 ± 2.1	73.59
Digital	89.99	-

4.4. Machine Learning Results

Having obtained the trained and fine-tuned models from the hardware-aware simulations, they were applied to the photonic tensor core inside the PyTorch framework. The main focus was set on the smaller initial CNN (conv01), for which the results are reported first. For both MNIST and CIFAR-10, experiments were conducted when either only the first convolutional layer, both convolutional layers, or all layers, including the fully connected one, were processed photonically. Afterwards, a brief study was made with the deeper and larger neural network containing four convolutional layers (conv02) with the goal of maximizing the performance on CIFAR-10. It was analyzed solely on that more complex dataset, with all its convolutional layers being processed on the PIC. The findings of this model are summarized in Section 4.4.2.

4.4.1. Performance of the Initial CNN

As with all subsequent experiments, four different models following the initial CNN architecture with two convolutional layers were tested: the purely digitally trained model, as well as the three previously introduced analog models with 10%, 50%, and 100% output noise. Due to time constraints, the models were not evaluated on the entire test set; however, the experiment was repeated five times on the same 500-image subset, grouped in a single batch. As a consequence, the mean accuracy still provides a reasonable estimate of the general performance, allowing for more tests to be carried out and enabling the observation of the photonic system's consistency. Additionally, training runs with five shuffled batches were conducted to increase the validity of the reported accuracies. Their plots can be found in the Appendix.

MNIST

All models were first tested with only their first layer being computed by the photonic tensor core. Figure 4.10 shows the outcome of the five test runs on the first 500 MNIST pictures and compares them to the digitally calculated accuracy for the same input. There are several important conclusions that can be drawn from the plot: First of all, the global accuracy is very high and always above 97%, but the analog models perform better than the original weights for low averaging. The blue curve, indicating the latter, increases eventually to peak values of more than 99% for avg $\geq 2^3$, which is as much as the digital calculation predicts. This implies that the stochastic effects in the system are no longer significant for this specific classification task. However, this is not surprising, as the noisy operations occurred only in the first layer, which is the least parameter-intensive part of the small CNN. Looking at the analog score progressions, the behavior

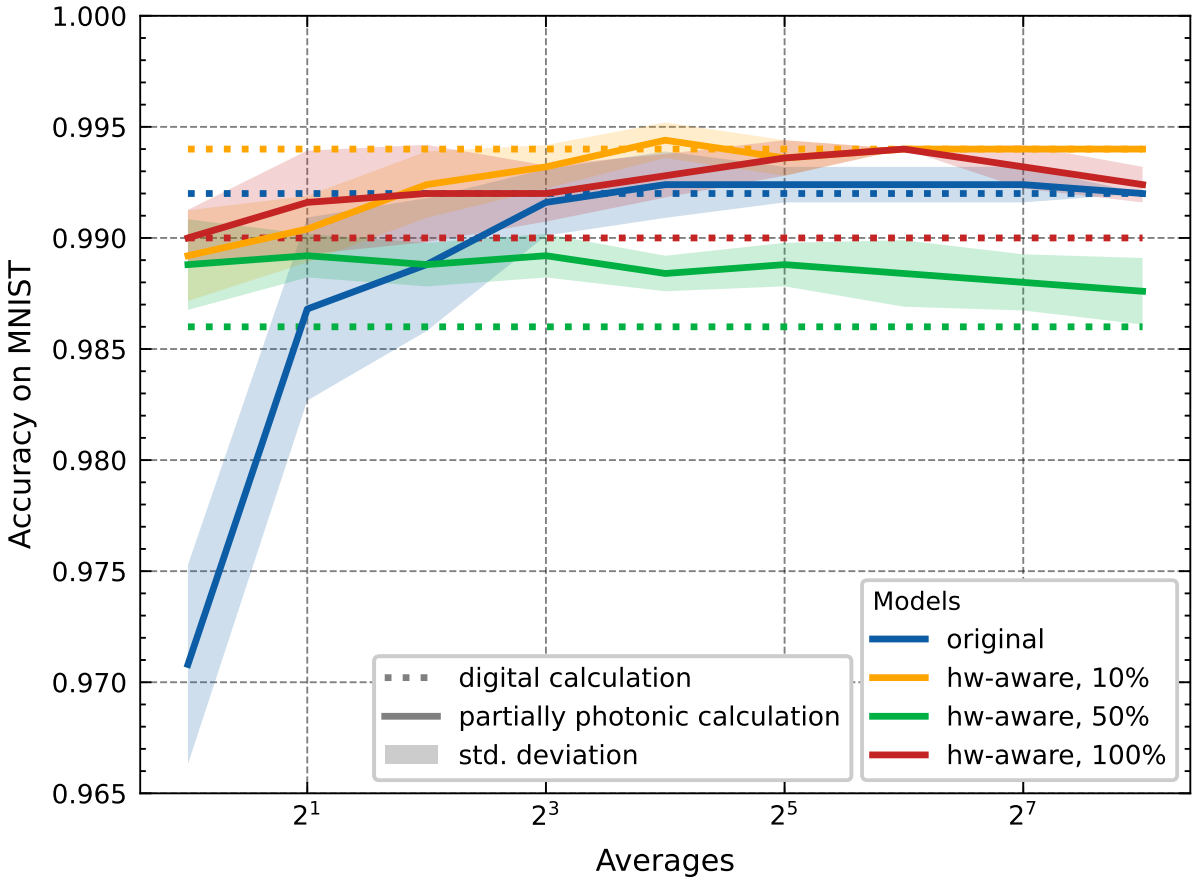


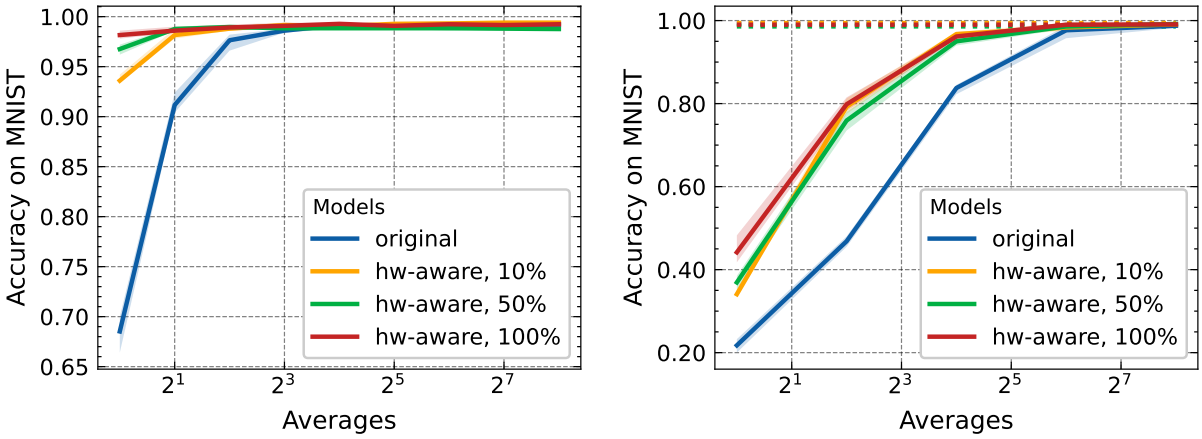
Figure 4.10.: MNIST accuracy for the first layer of the CNN conv01 being photonically calculated. Four different models were evaluated five times on a 500-picture batch from the test dataset, one model with conventionally trained weights (referred to as "original") and three models with finetuned weights obtained during the hardware-aware training. The latter were trained on three different noise settings inside AIHWKIT lightning. Only the first convolutional layer was processed on the optical system; the remaining convolutional layer and the fully connected one were computed digitally. The bold lines indicate the mean accuracy measured during the five tests; the shading in the background represents the standard deviation. The maximum potential of the different models—that is, if all layers were computed on a CPU with floating point precision—is visualized by the dotted lines (these plotting conventions are also valid in subsequent plots, which is why the additional legend will not be included every time). It can be seen that all models achieve high accuracies on MNIST, even for low averaging values. This is problematic since the minimum increase in accuracy is only 0.002 given the batch size, which is why differences between curves should not be overestimated. By utilizing the same batch in all runs, dataset variance is excluded, which is desired in this case. However, it also means that the maximum performances do not perfectly reflect the accuracies presented in the previous section (see Table 4.4), which were obtained on the entire dataset.

may seem unexpected at first: On the one side, the digital calculations (dotted lines) exceed the simulated accuracies reported in Table 4.4 and even change in order, with the model fine-tuned on 100% suddenly outperforming the model based on 50% output noise. Also, the digitally computed performances of the 10% noise level lie above the ones from the original, noise-free weights. On the other side, some photonic analog results surpass

their digital reference. These phenomena can be understood by taking the experimental procedure into account: Since only a subset of 500 instead of all 10,000 images is used, variations can occur. It seems like the chosen batch was coincidentally favorable for the 10% and the 100% analog CNNs, not for the intermediate model, though. Furthermore, the scale of the y-axis is very tight, such that the distances between curves do not actually correspond to significant performance differences. Recognizing correctly a single image more already causes an increase in accuracy of 0.002. Thus, seemingly surprising values can be explained with statistical fluctuations between different batches of the dataset and the low resolution of possible accuracy values. This is also a possible explanation why the green solid line, which depicts the partially photonic calculation, lies above the dotted digital reference line. Here, fewer than two images were additionally identified correctly on average. Nevertheless, the fact that this happened consistently in the case of the green line is unexpected and not fully understood yet. In following studies, more test runs could be performed to investigate this matter.
In general, we learn that MNIST is not the optimal evaluation metric, as all models score within a tight range, making the low resolution even more pronounced.

When the second convolutional layer was also photonically calculated, a better differentiation between models was observed. Figure 4.11a reveals, for instance, that the original weights now suffered significantly more from noise in the low-averaging region. Also, for no averaging, the analog models that were trained on more noise scored higher than less noise-robust ones. Since this setting is associated with the highest convolution errors of at least 60% (compare Figure 4.6), a differentiation between the analog models is expected, since all except the last analog model were fine-tuned under less noisy conditions. Conversely, this separation is no longer visible following the further progression of the curves, as all models can soon cope with the lower noise levels present during optical processing. As mentioned earlier, also test runs with five shuffled batches were conducted for the cases of one and two photonically calculated layers. For the latter case, the results on the MNIST set can be seen in Figure A.6. These curves show very similar performances and prove that the reported accuracies in this section are valid estimates for the entire data set.

The final MNIST tests applied both the convolution and MVM algorithms to perform all three layers, including the linear fully connected one, on the photonic tensor core. This means that all matrix-vector multiplications and convolutions necessary to predict labels within the artificial neural network were processed photonically; it is thus referred to as a fully photonic run. Only operations such as the bias addition or the activation function were still conducted in the digital domain. The results of three runs, again evaluating the same batch of handwritten digits, are shown in Figure 4.11b. Instead of testing all averaging values in the range from 2^0 to 2^8, only every second value was used for a test run due to time constraints. With all layers observing the analog non-idealities, the



(a) Two-layer photonic processing

(b) Fully photonic processing

Figure 4.11.: MNIST accuracies for the first two layers and all three layers being photonically computed. In a), both convolutional layers of the small CNN were processed on the PIC; for b), this also happened for the final linear layer, thus all convolutions and MVMs were processed on-chip (referred to as "fully photonic"). For b), only every second averaging setting was tested. Note the different scales of the y-axis, which have been chosen in order to better resolve details.

noise sensitivity increased notably compared to the previous two experiments. The analog models still outperform the original one, this time for the majority of averaging settings. But even the fine-tuned models do not achieve the maximum accuracies suggested by the hardware-aware simulations before averaging reaches values of 2^5 and higher. Moreover, the distinction between the different analog models becomes less emphasized, with the 10% and the 50% model both scoring almost exactly the same. This is likely a consequence of the MVM processing, which happens within the FC layer at the end of the CNN and is responsible for generating the actual predictions. Firstly, the MVM error has already been shown to be notably higher for lower averaging than the deviation observed with the convolution algorithm (see Figure 4.3), due to advanced schemes incorporated within the latter. In addition to that, the operations inside a linear layer, even if it just consists of ten neurons as in this case, require an extensive use of weight matrix tiling, which leads to a manifold accumulation of noise. For MNIST, the conv01 architecture stores 15680 of its parameters alone inside the weights of the last layer, and the MVM algorithm has to process an operation of shape $(500 \times 1568) \cdot (1568 \times 10)$ for a batch containing 500 images. With the current method of excluding the broken channel that cannot be set to zero, the PIC receives only five weights per time step, meaning that at least 3,136 weight tiles are required, with the actual number being even slightly higher because of the implemented mechanism. Since the output of the FC layer is simply a ten-dimensional vector, the results from hundreds of these atomic MVM operations are added to obtain a single scalar. If all noise on the single tiles were uncorrelated, then it should not matter how many tiles are added. However, since it is possible that there are also systematic, correlated error sources, such as a slight bias in the weight setting

or scaling, this noise may accumulate. This may prevent accurate classification in many cases and further undermine the robustness against output and weight noise obtained from the AIHWKIT-Lightning simulations.
Still, for high averaging settings, all models achieved accuracies of at least 99.0%.

CIFAR-10

In a parallel fashion, the small CNN was investigated when assigned the CIFAR-10 dataset. In the first experiment, only the first convolutional layer was processed photonically again, shown in Figure 4.12. The result differs from the corresponding figure of the MNIST analysis, especially because the distinct models can be differentiated very clearly. As outlined already, the CIFAR-10 benchmark is much more complex and consists a multiple of pixel data per image compared to MNIST, which is why classification is generally more difficult, especially for small neural networks such as the current one. The simulations in Section 4.3 have shown that the overall accuracy is lower on this dataset, but also that the different analog models differ more strongly, with higher noise robustness automatically leading to a sacrifice of potential optimal accuracy. For CIFAR-10, the simulated accuracies span a range from approximately 57% to 67% while the values varied less than 1.5% for MNIST (see Table 4.4). The enlarged range of accuracies allows differences to be resolved very well, as the accuracy resolution of 0.2% is small compared to the entire range. The splitting occurs among the digital reference calculations (dotted lines) as well as among the partially photonically obtained model performances (solid lines). Similarly to the MNIST case, the digital accuracy is not exactly the same as that found in the simulations, but this discrepancy is again explained by the fact that this reference was also obtained on only a single test batch. However, the noise-dependent order of maximum performances was now very well conserved, especially when looking at the performances for high averaging. The evaluation of the original model shows that the non-adjusted weights only function effectively for medium to high averaging, but for avg $\geq 2^6$ they surpassed all other models and reached their digitally predicted optimum at avg $= 2^8$. For no averaging at all, both the 50% and the 100% noise analog models scored equally well and correctly classified already about six out of ten images, which means that the amount of noise present at this averaging setting could be handled just as well by the in theory less robust model. Here, it is important to note that the optical noise only affected the computations in the first layer, whereas the analog models were trained with all MVMs being non-ideal. Thus, it is not necessarily surprising that the 50% model was as good as the 100% model for avg $= 2^0$.

The further experiments examined the inference capabilities of all models when the two convolutional layers and when all three layers of conv01 were processed photonically. The

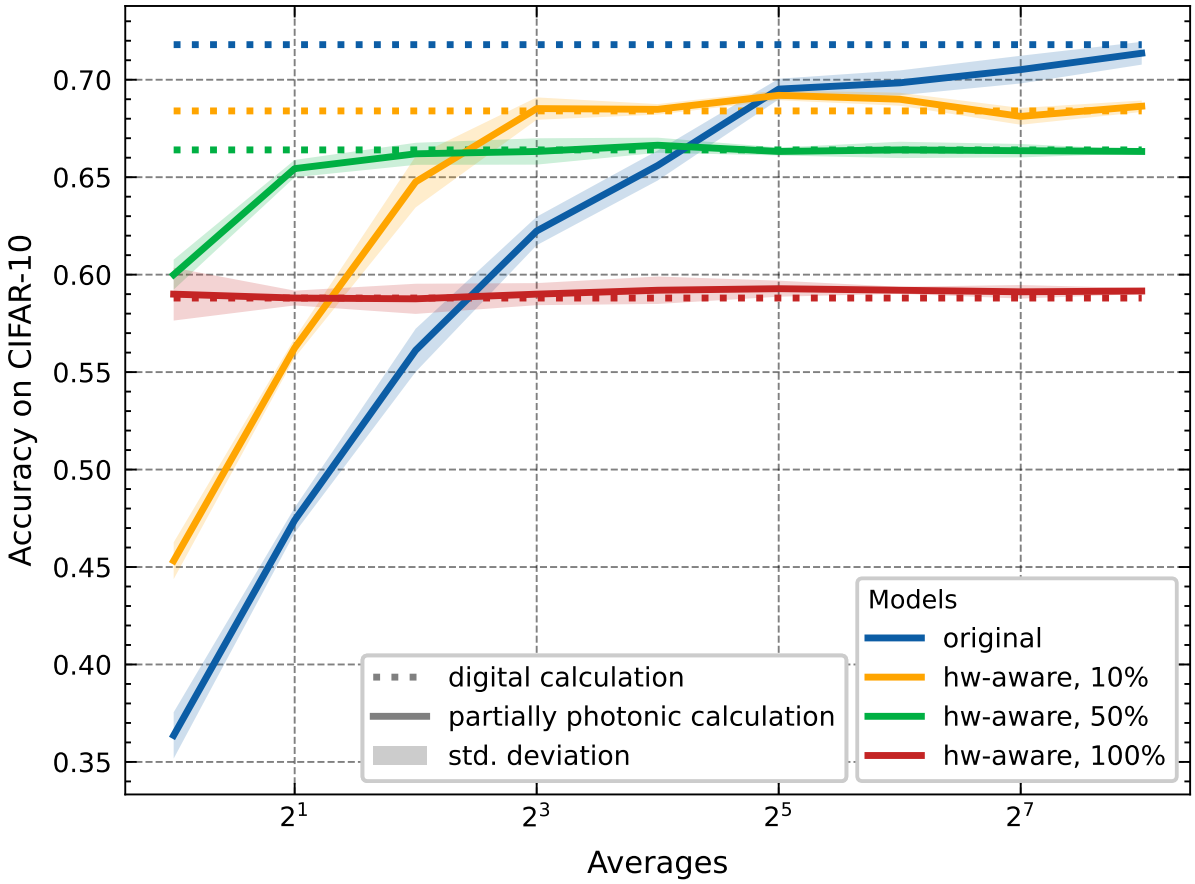
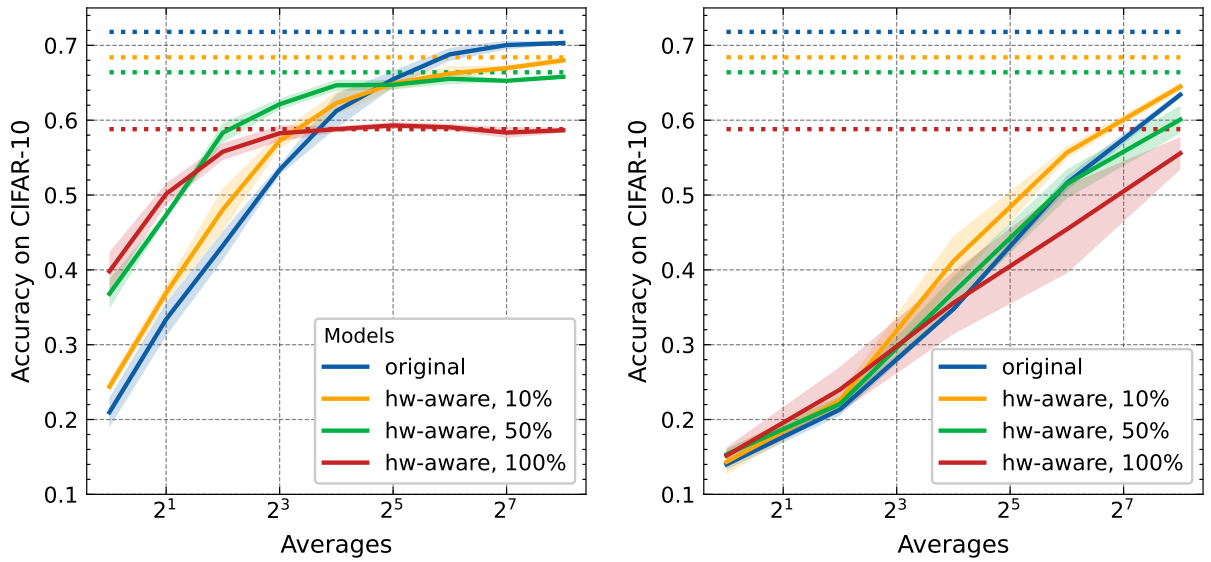


Figure 4.12.: CIFAR-10 accuracy for the first layer of the CNN conv01 being photonically calculated. The four different versions of the shallow convolutional neural network were tested five times on the same 500-image subset, with each time only the first layer being operated by the photonic convolution algorithm, while subsequent layers were processed with floating point accuracy.

results can be found in Figures 4.13a and 4.13b, respectively. In the former, we observe the split-up of the models again, but in a less pronounced way. Still, for high averaging, a clear order of the maximum accuracy according to the noise level of the models is apparent. In the latter, this is not the case, and the different models can hardly be distinguished, especially for low averaging amounts. As already noticed in Figure 4.11b, the curves also do not increase as strongly as in the cases with digitally computed FC layers. This implies once more that the inclusion of the MVM algorithm in these models renders the fine-tuning of the models less effective. As mentioned before, the MVM error is generally worse compared to the convolution error, and the large number of weights poses a challenge for the photonic system because extensive tiling is required, given the size of the crossbar array. For the CIFAR-10 adjusted models, the linear layer consists of about 30% more weights than the MNIST models, which intensifies the tiling problem if uncorrelated noise accumulates. However, further investigations need to be conducted to fully understand the observed effect.



(a) Two-layer photonic processing

(b) Fully photonic processing

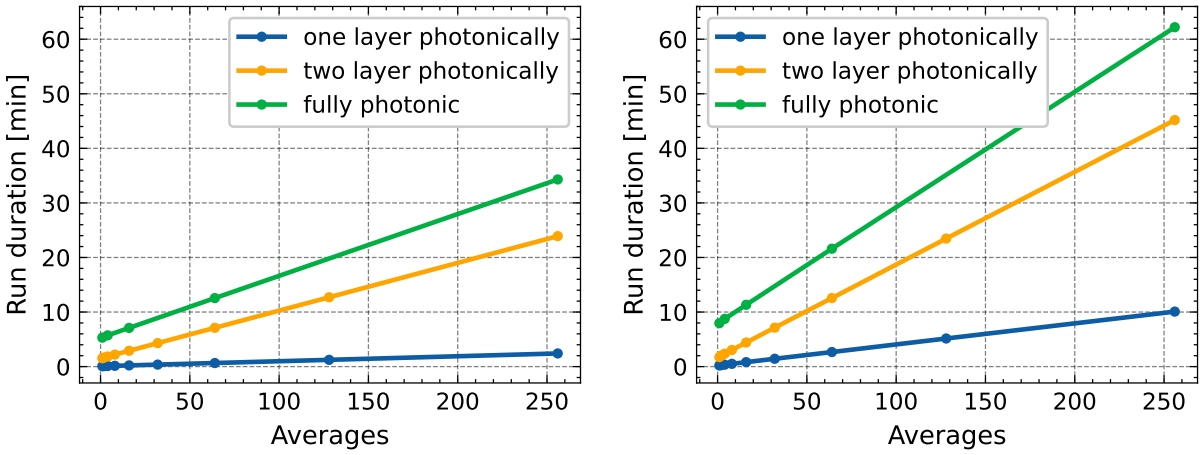
Figure 4.13.: CIFAR-10 accuracies for the first two layers and all three layers being photonically computed. For (a), five test runs were conducted per model on the same batch, whereas three runs each were analyzed in (b) for every second averaging value.

Nevertheless, both the convolution and the MVM program perform in their respective roles, enabling partially as well as fully photonic machine learning runs, even though the latter require higher averaging to achieve comparable results. Especially the deployment of the photonic convolutions has proven to work well.

In the supplementary Figure A.7, the performances of models with two photonic layers can be found, which were evaluated on shuffled batches. As in the case of MNIST, the accuracies align with the ones reported here, but naturally exhibit stronger fluctuations.

Process Duration

During all conducted experiments on the photonic system, the time consumption of the test runs was recorded. In Figure 4.14, the processing times of MNIST and CIFAR-10 are shown over varying averaging settings and separated by whether just one, the first two, or all three layers of the small CNN were computed on the photonic tensor core. As expected, higher averaging requires proportionally more time resources, just as more optical processing does. Models aiming to classify CIFAR-10 images generally observe longer run times, which is a consequence of the increased image size and the three RGB channels. A notable offset between the run times of fully photonically calculated models and the other ones indicates the impact of the substantial weight switching processes required in the FC layer. As discussed in Section 3.1, the processing speed of the current experimental setup is limited by various factors, such as the small size of the usable crossbar array or



(a) MNIST run times

(b) CIFAR-10 run times

Figure 4.14.: Run times of partially and fully photonically calculated models over varying average settings. For these curves, the durations of five inference runs of the CNN (conv01) over 500-image batches were averaged for MNIST (a) and CIFAR-10 (b); the standard deviations indicated by colored shading are vanishing and thus not visible. Note that the x-axis scales linearly, in contrast to previous plots, to emphasize that the time consumption does so too. The increased duration of CIFAR-10 runs is due to the larger image size and RGB channels.

the data transfer rates currently possible. It should be noted therefore that the reported times of Figure 4.14 are very specific to this system in its prototype state and by no means fundamental.

4.4.2. Deployment of a Deeper CNN

In the final step, the deeper convolutional neural network and its model variations were evaluated for selected averaging settings, which stems from the long run times encountered with this larger architecture. Each time, all convolutional layers were processed with the PIC, while the MVMs occurring in the FC layer were calculated digitally. The experiment included three different parts: Firstly, the low averaging settings avg $= 2^0$ and avg $= 2^2$ were assessed with the model fine-tuned to cope with 100% output noise to gain information about potential performance with significant convolution errors. Then, a measurement sweep was performed for the setting avg $= 2^5$, testing all four models at a medium noise level. Finally, one single run of the analog model trained with 10% noise was conducted for avg $= 2^6$, aiming to achieve an overall maximum accuracy on CIFAR-10. All results are reported in Table 4.6. The low-averaging analysis indicated that some amounts of noise inside the system cannot be handled even by the very robust analog model, scoring only 16.8% without any averaging. However, already the setting avg $= 2^2$ drastically improved the models' performance.
For medium averaging values, the analog models achieved accuracies not possible with

the smaller CNN, even when it was processed completely digitally. Furthermore, at least for this specific setting, the fine-tuning notably improved the original model; in the case of the 10% output noise model, an increase of more than 12 percentage points was exhibited. As a consequence, the 10% model has been chosen for a single run with even higher averaging. This resulted in 85.0% accuracy, the best photonically achieved performance on CIFAR-10 observed in this study. This value lies just 1.1 percentage points below the digitally calculated optimum (compare Table 4.5). Although it required a substantial runtime under current conditions, it is a clear demonstration of the practical feasibility of multi-layer neural network applications on photonic tensor cores.

Table 4.6.: Deep CNN performances on CIFAR-10. The data was obtained from single test runs on the first 500 images from the dataset, with all four convolutional layers being processed photonically and only the FC layer, consisting of ten neurons, being calculated in the digital domain.

Model	Averaging	CIFAR-10 Accuracy [%]	Run Time [min]
Low averaging evaluation of the 100% analog model			
HW-aware, 100%	2^0	16.8	45.59
HW-aware, 100%	2^2	56.0	63.24
Medium averaging evaluation of all model variations			
Original	2^5	67.8	201.47
HW-aware, 10%	2^5	80.0	201.33
HW-aware, 50%	2^5	79.0	201.61
HW-aware, 100%	2^5	72.6	199.80
High averaging evaluation of the 10% analog model			
HW-aware, 10%	2^6	85.0	358.04

5. Conclusion

In this Bachelor's thesis, control algorithms for an analog photonic integrated system with a photonic tensor core have been developed with the objective of computing matrix-vector multiplications and two-dimensional discrete convolutions for applications in machine learning. After a hardware-aware training, two different convolutional neural network architectures were deployed on the setup based on these new algorithms, and they could perform popular image recognition tasks, such as the MNIST and CIFAR-10 benchmarks, with some or all of the necessary MVMs and convolutions being optically processed.

The components and operation of the photonic integrated system at hand were explained while outlining general challenges of analog photonic computations, such as non-linear weight scaling, channel-wise inhomogeneities, and electrical crosstalk. To account for these non-idealities, a weight setting scheme has been developed that effectively captures relevant system behavior inside a weight map and applies this to set weights in a reliable and accurate manner. Especially important is that the new scheme performed the setting process two orders of magnitude faster than the previous iterative method, laying the basis for machine learning applications that require vast amounts of distinct weights. By deploying an automatic crosstalk measurement and correction mechanism, the precision of the truly set weights could be further improved. Afterwards, specialized algorithms were created that utilized the existing software interface as well as the new weight setting scheme to perform MVMs and convolution operations photonically on-chip. For this purpose, an output scaling method was introduced that operates solely with the weight map information and does not require digitally computed reference matrix multiplications. This scaling achieves an almost optimal transformation of the physical output range into an informative, absolute scale.
The performances of the new algorithms were analyzed over various averaging settings, illuminating the correlations between the amount of averaging, the MVM and convolution errors, and the time consumption. Without any averaging, errors ranged from about 60% to 100%, but higher averaging drastically improved these values up to medium single-digit numbers. The experiments showed that while averaging provides a well-controllable determinant of the output accuracy, the calculation errors cannot be fully eliminated.

To account for imperfect calculations and other hardware constraints, such as quantization, in machine learning tasks, the methodology of hardware-aware training was pre-

sented. In detail, the software library AIHWKIT-Lightning was used to fine-tune conventionally pretrained neural networks under the consideration of analog hardware characteristics, thereby especially increasing the model's robustness towards input and weight noise. Two simple convolutional neural network architectures were designed to classify images from the MNIST and CIFAR-10 datasets, with one network including two sequential convolutional layers and the other four, followed by a ten-neuron linear layer in both cases. During fine-tuning, three distinct output noise levels were applied, oriented to the range observed in the evaluation of the control algorithms. The fine-tuning on noise significantly improved all models in their simulated inference capability in comparison to non-adjusted models, even though higher robustness came at the cost of lower maximum accuracy.

Finally, the hardware-aware models were deployed on the experimental setup to perform actual machine learning tasks with the photonic tensor core. The shallow CNN has been tested for either one, two, or three of its layers being calculated optically, achieving over 99% accuracy on a random 500-image subset of MNIST for all cases. When all MVMs and convolutions were processed photonically, this required higher averaging, but when exclusively the convolutional layers were computed in the analog domain, maximum scores were already achieved for averaging amounts of avg = 2^2. The hardware-aware models proved to be significantly better in the high-noise regime than the original model, underscoring the importance of fine-tuning. The MNIST dataset, known for its simplicity, did not allow for a strong distinction between differently trained analog models, since their maximum performances were very close to each other. For CIFAR-10, posing a much more complex challenge to the models, a clear distinction could be observed: More robust models scored better for high noise levels, but they were surpassed by less robust ones as noise decreased with averaging. In the case of the two-layer photonic processing, accuracies of up to 70.0% were measured, which was only 1.2 percentage points less than the digitally computed maximum. When the linear layer was also calculated photonically, the performances decreased and the splitting of noise levels became less pronounced. This is most likely caused by a combination of higher noise sensitivity in MVM calculations compared to convolutions and the excessive amount of tiling required due to the given size limitation of the crossbar array, which could lead to an accumulation of slight systematic errors. To fully understand this phenomenon, it should be further analyzed, though. Still, a maximum score of 65% was observed.

The analysis with the larger CNN demonstrated successfully the concatenation of four photonically processed convolutional layers, paving the way for deep learning applications. In the best test run, the model fine-tuned for 10% output noise achieved an accuracy of 85.0% on the CIFAR-10 benchmark, only 1.1 percentage points below an ideal digital outcome.

As discussed in chapter 3, the processing speed of the system is currently strongly limited by the electronic infrastructure and the small size of the usable photonic tensor core. Because of this, inference runs still last for minutes or even hours. However, the speed constraints primarily affect the weight switching, while vast amounts of input data can already be processed efficiently. For instance, a convolution layer in an MNIST model leverages the system's capabilities best for batch sizes of at least 500 images, pointing out the potential of optical hardware accelerators for use cases with high data throughput.

In conclusion, the newly developed methods have enabled machine learning applications on the present photonic system. The introduced algorithms provide a flexible approach to mitigate common non-idealities and can be used in future works as well. While not achieving actual time savings yet, this study serves as proof of concept for the deployment of deep learning on photonic tensor cores.

Outlook

Throughout the investigations, numerous options for potential improvements have been identified. For example, utilizing other light sources and optical amplifiers that exhibit less power fluctuation is a promising way to reduce the necessary amount of averaging. This could be realized with lasers instead of ASE devices. Concerning the training of the neural networks, an analysis of how exactly the hardware-aware fine-tuning transforms the model parameters could provide valuable insights on how neural networks should be designed in the first place to suit analog processing circumstances. Also, accounting for the tiling during the AIHWKIT-Lightning simulations may increase the performance of fully photonically computed inference runs. Additionally, subsequent studies may benefit notably from a thorough hyperparameter search for both pre-training and fine-tuning, which can help determine the optimal learning rate or amount of regularization, for instance.

In future iterations of the PIC, we can expect larger crossbar arrays, which will both increase data throughput and reduce tiling requirements. As also mentioned in the theoretical background, wavelength division multiplexing (WDM) is an auspicious approach to achieve parallelization exclusive to the photonic computation domain.

For now, the introduced methods can serve as a starting point for manifold experiments regarding photonic machine learning or other applications that utilize optically calculated MVMs and convolutions.

Bibliography

[1] AMD. *AMD Zynq UltraScale+ RFSoC ZCU216 Evaluation Kit*. Accessed on 06/03/2025. URL: https://www.amd.com/en/products/adaptive-socs-and-fpgas/evaluation-boards/zcu216.html.

[2] J. Rasmus Bankwitz et al. "Towards smart transceivers in FPGA-controlled lithium-niobate-on-insulator integrated circuits for edge computing applications". In: *Opt. Mater. Express* 13.12 (Dec. 2023), pp. 3667–3676. DOI: 10.1364/OME.503340. URL: https://opg.optica.org/ome/abstract.cfm?URI=ome-13-12-3667.

[3] Marlon Becker, Dominik Drees, and Benjamin Risse Frank Brückerhoff-Plückelmann Carsten Schuck2 Wolfram Pernice. "Activation Functions in Non-Negative Neural Networks". In: *Neural Information Processing Systems* (2023).

[4] Lucía Bouza, Aurélie Bugeau, and Loïc Lannelongue. "How to estimate carbon footprint when training deep learning models? A guide and review". In: *Environmental Research Communications* 5.11 (Nov. 2023), p. 115014. ISSN: 2515-7620. DOI: 10.1088/2515-7620/acf81b. URL: https://iopscience.iop.org/article/10.1088/2515-7620/acf81b.

[5] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[6] J. Buchel et al. "Gradient descent-based programming of analog in-memory computing cores". In: *2022 International Electron Devices Meeting (IEDM)*. San Francisco, CA, USA: IEEE, Dec. 2022, pp. 33.1.1–33.1.4. ISBN: 9781665489591. DOI: 10.1109/IEDM45625.2022.10019486. URL: https://ieeexplore.ieee.org/document/10019486/ (visited on 06/04/2025).

[7] Julian Büchel et al. "AIHWKIT-Lightning: A Scalable HW-Aware Training Toolkit for Analog In-Memory Computing". In: *NeurIPS 2024 Workshop Machine Learning with new Compute Paradigms*. 2024. URL: https://openreview.net/forum?id=QNdxOgGmhR.

[8] Jan Chorowski and Jacek M. Zurada. "Learning Understandable Neural Networks With Nonnegative Weight Constraints". In: *IEEE Transactions on Neural Networks and Learning Systems* 26.1 (2015), pp. 62–69. DOI: 10.1109/TNNLS.2014.2310059.

[9] Frédérique Deshours et al. "New Nonlinear Electrical Modeling of High-Speed Electroabsorption Modulators for 40 Gb/s Optical Networks". In: *Journal of Lightwave Technology* 29.6 (2011), pp. 880–887. DOI: 10.1109/JLT.2011.2106110.

[10] Bowei Dong et al. "Partial coherence enhances parallelized photonic computing". In: *Nature* 632.8023 (Aug. 1, 2024), pp. 55–62. ISSN: 1476-4687. DOI: 10.1038/s41586-024-07590-y. URL: https://doi.org/10.1038/s41586-024-07590-y.

[11] Karl Joachim Ebeling. *Integrierte Optoelektronik*. de. Berlin, Heidelberg: Springer Berlin Heidelberg, 1992. ISBN: 9783540546559 9783662079454. DOI: 10.1007/978-3-662-07945-4. URL: http://link.springer.com/10.1007/978-3-662-07945-4.

[12] Konrad Engel. *Mathematische Grundlagen des überwachten maschinellen Lernens: Optimierungstheoretische Methoden*. de. Berlin, Heidelberg: Springer Berlin Heidelberg, 2024. ISBN: 9783662681336 9783662681343. DOI: 10.1007/978-3-662-68134-3. URL: https://link.springer.com/10.1007/978-3-662-68134-3.

[13] William Fedus, Barret Zoph, and Noam Shazeer. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. 2022. arXiv: 2101.03961 [cs.LG]. URL: https://arxiv.org/abs/2101.03961.

[14] J. Feldmann et al. "Parallel convolutional processing using an integrated photonic tensor core". In: *Nature* 589.7840 (Jan. 1, 2021), pp. 52–58. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03070-1. URL: https://doi.org/10.1038/s41586-020-03070-1.

[15] Alexander L. Fradkov. "Early History of Machine Learning". In: *IFAC-PapersOnLine* 53.2 (2020). 21st IFAC World Congress, pp. 1385–1390. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2020.12.1888. URL: https://www.sciencedirect.com/science/article/pii/S2405896320325027.

[16] Jonathan K. George et al. "Neuromorphic photonics with electro-absorption modulators". In: *Opt. Express* 27.4 (Feb. 2019), pp. 5181–5191. DOI: 10.1364/OE.27.005181. URL: https://opg.optica.org/oe/abstract.cfm?URI=oe-27-4-5181.

[17] John L. Hall. "Nobel Lecture: Defining and measuring optical frequencies". In: *Rev. Mod. Phys.* 78 (4 Nov. 2006), pp. 1279–1295. DOI: 10.1103/RevModPhys.78.1279. URL: https://link.aps.org/doi/10.1103/RevModPhys.78.1279.

[18] Salman Khan et al. *A guide to Convolutional Neural Networks for Computer Vision*. Springer Cham, Jan. 1, 2018. DOI: 10.1007/978-3-031-01821-3. URL: https://doi.org/10.1007/978-3-031-01821-3.

[19] Marc J. Madou. *Manufacturing Techniques for Microfabrication and Nanotechnology*. en. 0th ed. CRC Press, June 2011. ISBN: 9780429112461. DOI: 10.1201/9781439895306. URL: https://www.taylorfrancis.com/books/9781439895306.

[20] Mario Miscuglio and Volker J. Sorger. "Photonic tensor cores for machine learning". In: *Applied Physics Reviews* 7.3 (July 2020), p. 031404. ISSN: 1931-9401. DOI: 10.1063/5.0001942. eprint: https://pubs.aip.org/aip/apr/article-pdf/doi/10.1063/5.0001942/14577276/031404_1_online.pdf. URL: https://doi.org/10.1063/5.0001942.

[21] Richard Osgood and Xiang Meng. *Principles of Photonic Integrated Circuits: Materials, Device Physics, Guided Wave Design*. en. Graduate Texts in Physics. Cham: Springer International Publishing, 2021. ISBN: 9783030651923 9783030651930. DOI: 10.1007/978-3-030-65193-0. URL: https://link.springer.com/10.1007/978-3-030-65193-0.

[22] Alexandre Piquard. "The explosion in AI-related electricity demand has already had local consequences". In: *Le Monde* (2024). Accessed on 06/04/2025.

[23] Malte J. Rasch et al. "Hardware-aware training for large-scale and diverse deep learning inference workloads using in-memory computing-based accelerators". en. In: *Nature Communications* 14.1 (Aug. 2023), p. 5282. ISSN: 2041-1723. DOI: 10.1038/s41467-023-40770-4. URL: https://www.nature.com/articles/s41467-023-40770-4 (visited on 06/04/2025).

[24] Patrick Schnabel. *Kommunikationstechnik-Fibel: Grundlagen der Kommunikationstechnik, Netze, Funktechnik und Mobilfunk, Breitbandtechnik, Voice over IP*. ger. 5. vollständig überarbeitete Auflage. Ludwigsburg: Schnabel, 2019. ISBN: 9783833005671.

[25] Yichen Shen et al. "Deep learning with coherent nanophotonic circuits". en. In: *Nature Photonics* 11.7 (July 2017), pp. 441–446. ISSN: 1749-4885, 1749-4893. DOI: 10.1038/nphoton.2017.93. URL: https://www.nature.com/articles/nphoton.2017.93 (visited on 06/13/2025).

[26] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[27] Irwin Sobel. "An Isotropic 3x3 Image Gradient Operator". In: *Presentation at Stanford A.I. Project 1968* (Feb. 2014).

[28] Alexander N. Tait et al. "Neuromorphic photonic networks using silicon photonic weight banks". en. In: *Scientific Reports* 7.1 (Aug. 2017), p. 7430. ISSN: 2045-2322. DOI: 10.1038/s41598-017-07754-z. URL: https://www.nature.com/articles/s41598-017-07754-z.

[29] Apostolos Tsakyridis et al. "Photonic neural networks and optics-informed deep learning fundamentals". In: *APL Photonics* 9.1 (Jan. 2024), p. 011102. ISSN: 2378-0967. DOI: 10.1063/5.0169810. eprint: https://pubs.aip.org/aip/app/article-pdf/doi/10.1063/5.0169810/19848606/011102_1_5.0169810.pdf. URL: https://doi.org/10.1063/5.0169810.

[30] Nicolas Valero et al. "High-power amplified spontaneous emission pulses with tunable coherence for efficient non-linear processes". en. In: *Scientific Reports* 11.1 (Mar. 2021), p. 4844. ISSN: 2045-2322. DOI: 10.1038/s41598-021-83443-2. URL: https://www.nature.com/articles/s41598-021-83443-2 (visited on 06/03/2025).

[31] Jeremy Watt, Aggelos Konstantinos Katsaggelos, and Reza Borhani. *Machine learning refined: foundations, algorithms, and applications*. eng. Cambridge: Cambridge university press, 2016. ISBN: 9781107123526.

[32] H. Yamada et al. "Optical directional coupler based on Si-wire waveguides". In: *IEEE Photonics Technology Letters* 17.3 (2005), pp. 585–587. DOI: 10.1109/LPT.2004.840926.

[33] Xia Zhao et al. "A review of convolutional neural networks in computer vision". In: *Artificial Intelligence Review* 57.4 (Mar. 23, 2024), p. 99. ISSN: 1573-7462. DOI: 10.1007/s10462-024-10721-6. URL: https://doi.org/10.1007/s10462-024-10721-6.

[34] Hailong Zhou et al. "Photonic matrix multiplication lights up photonic accelerator and beyond". en. In: *Light: Science & Applications* 11.1 (Feb. 2022), p. 30. ISSN: 2047-7538. DOI: 10.1038/s41377-022-00717-8. URL: https://www.nature.com/articles/s41377-022-00717-8.

A. Appendix

Additional Figures

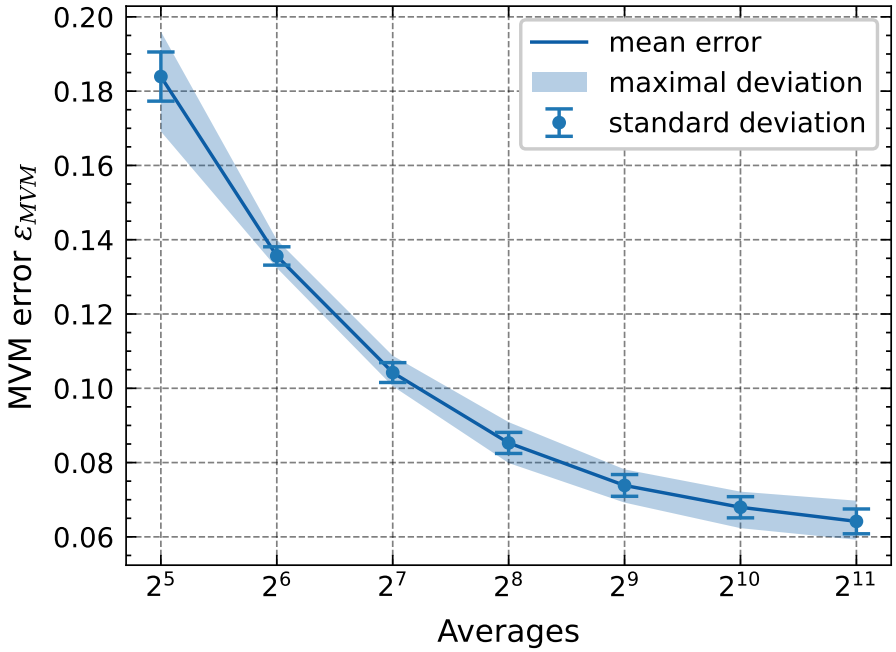
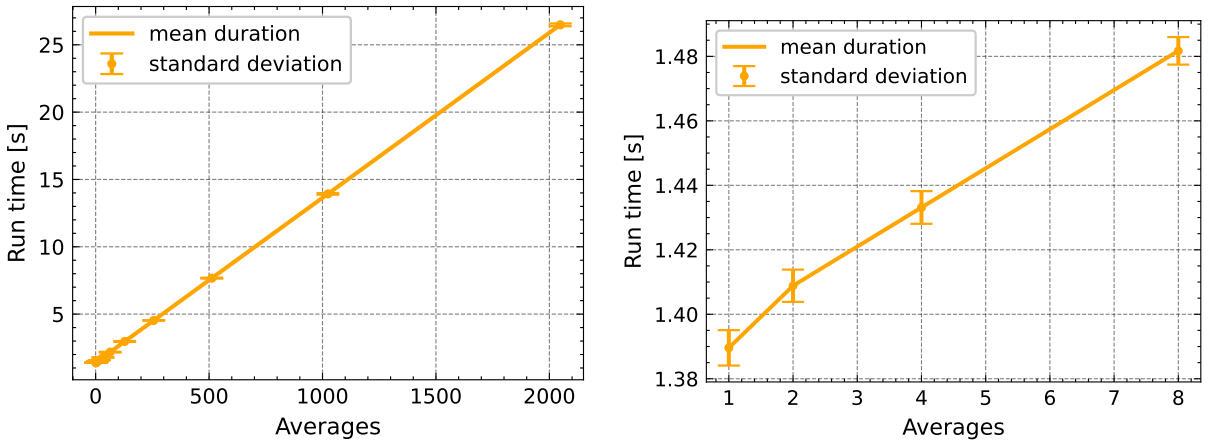


Figure A.1.: MVM error detail for higher averaging settings. This plot is a zoomed-in representation of Figure 4.3a, showing the errors of stronger averaging in greater resolution.



(a) Entier averaging range **(b)** Detail view

Figure A.2.: Time consumption of MVM benchmark. These plots show that the time of the MVM operation scales linearly with the averaging, as expected. Compare Figure 4.3b.

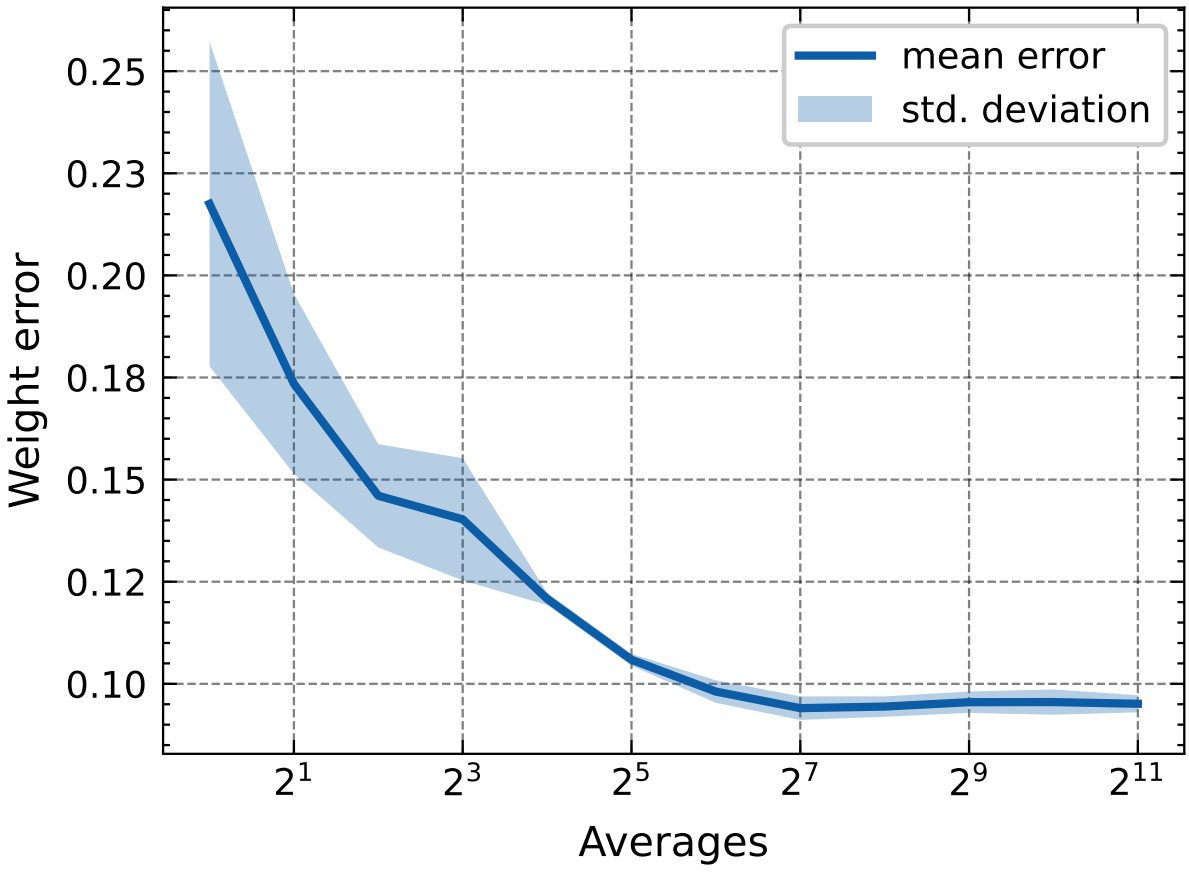


Figure A.3.: Weight error ϵ_{weight} of the iterative weight setting algorithm when only benign weights are assigned. This plot was created similarly to the plots of Figure 4.5, but this time the iterative algorithm only received weights that match well the channel inhomogenities. Suddenly, this scheme performs tremendously better compared to the case of random weights; it is also more stable and competitive with the newly introduced methods in terms of accuracy under these circumstances.

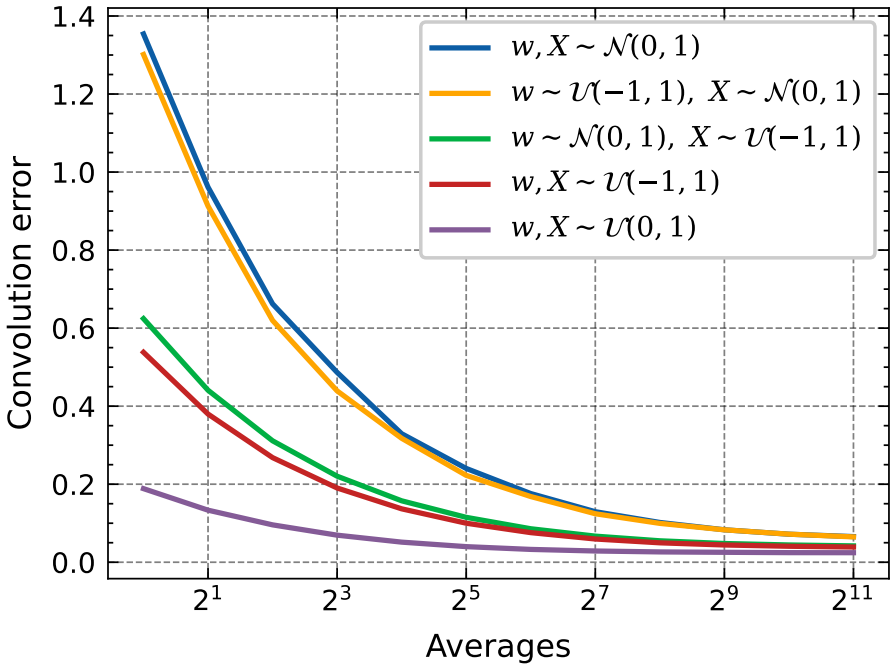
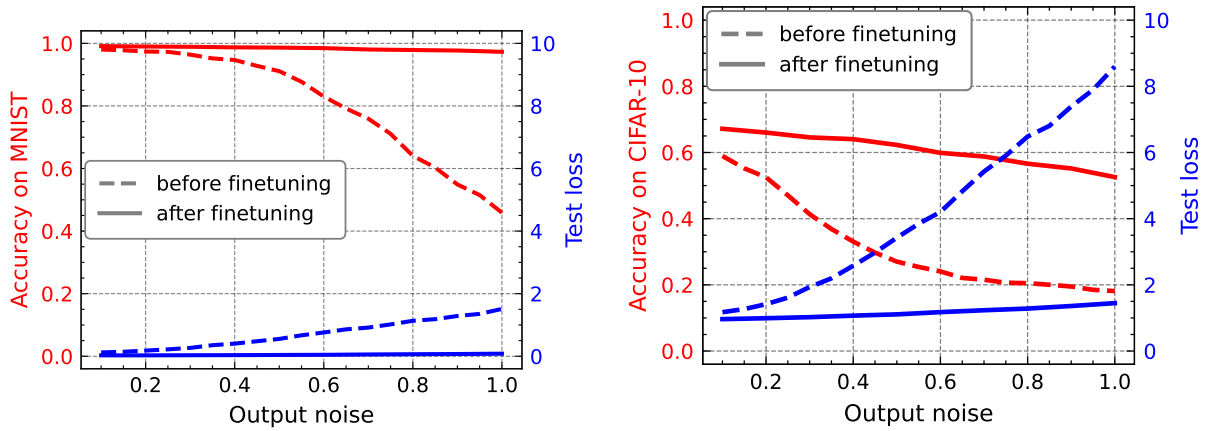
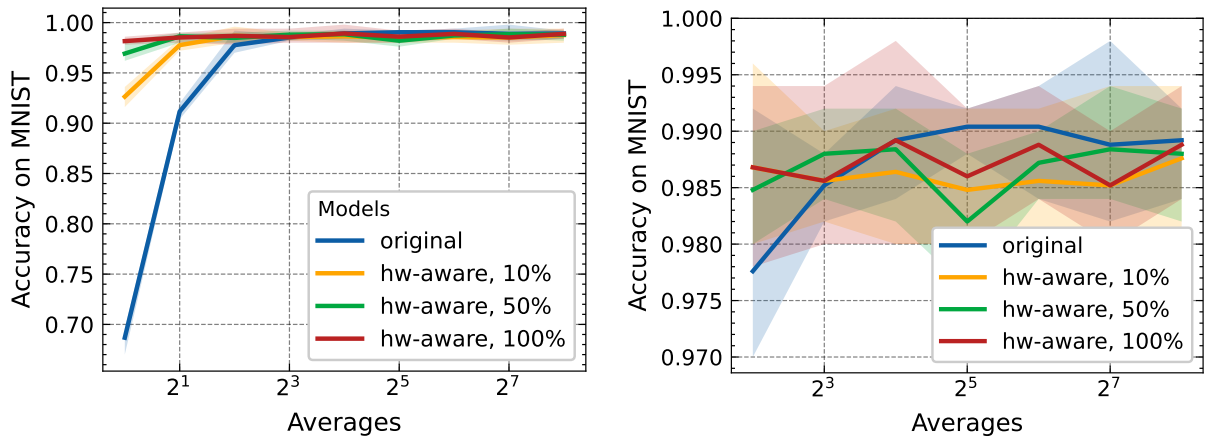


Figure A.4.: Impact of input and weight distributions on the convolution error. The errors were obtained, just like for the data of Figure 4.6, by the repeated convolution of a tensor of shape [100, 1, 28, 28]. The distribution, from which the values for the input and the weight tensors were drawn, has a strong impact on the performance of the photonic computation results. Generally, the highest error metrics are associated with normally distributed inputs, while uniformly distributed ones perform better. The errors can be further decreased by sampling inputs and weights from positive-only uniform distributions. Colored shadings or error bars, indicating the standard deviation, have been left out for greater visual clarity.



(a) MNIST benchmark **(b)** CIFAR-10 benchmark

Figure A.5.: Noise-dependent inference capability of analog models before and after finetuning within the AIHWKIT-Lightning simulation. For this test, the pre-trained digital model (conv01) was converted into analog models of various noise levels. The performance of these models was measured immediately after the conversion, and once again after each model had been trained in a hardware-aware manner for ten iterations. It is apparent that the fine-tuning has a significant impact on the accuracy, improving it drastically, especially for larger amounts of noise. However, the maximum digital accuracy reported in Section 4.3 was not quite reached again. As expected, increased robustness to noise comes at the price of decreased optimal performance.



(a) Entier averaging range **(b)** Detail view

Figure A.6.: Conv01 performance on MNIST for two photonically calculated convolutional layers when five shuffled batches were evaluated. The data was obtained in a similar way as for Figure 4.11a, but the five test runs were conducted on five shuffled 500-image batches.

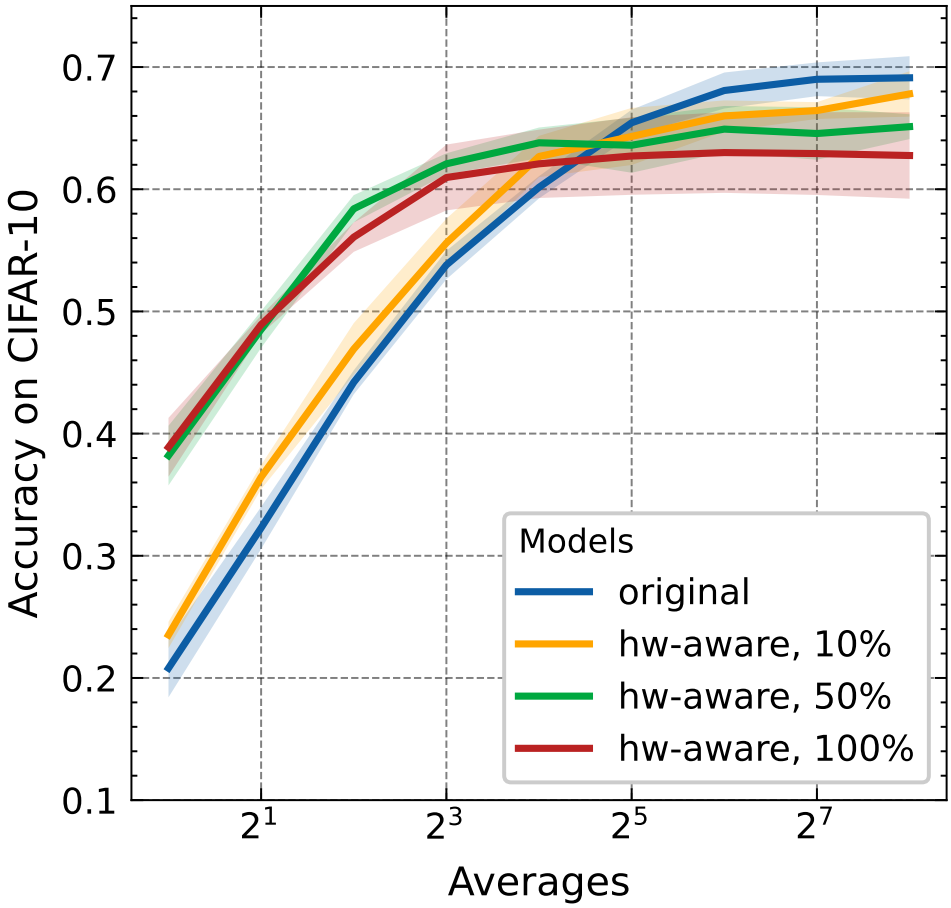


Figure A.7.: Conv01 performance on CIFAR-10 for two photonically calculated convolutional layers when five shuffled batches were evaluated. Just like in Figure 4.13a, five test runs were conducted per model, but here five shuffled batches with 500 images each were evaluated.

Acknowledgements

At the end of this thesis, I want to thank Prof. Dr. Wolfram Pernice sincerely for giving me the opportunity to join his research group, Neuromorphic Quantumphotonics, and to work on this exciting project under his supervision. Additionally, I would like to thank Prof. Dr. Holger Fröning for agreeing to be my second examiner.

I would like to extend my deepest gratitude to my supervisor, Jelle Dijkstra, who has guided me from my very first day on the project and provided me with countless valuable pieces of advice. He was always approachable and also revised my report in multiple stages. Furthermore, he developed the photonic integrated system, including its hardware and software interfaces, which made this work possible in the first place.

I am very thankful for the advice I received from Frank Brückerhoff-Plückelmann, who shared his extensive experience and insights whenever I had questions. I also want to thank him for revising the thesis.

Special thanks are due to my friends Christoph Hillgruber and Pablo Miltenyi Martinez, who offered to proofread this work and provided me with numerous helpful propositions.

I also want to thank all members of the Neuromorphic Quantumphotonics group who welcomed me very warmly and introduced me to experimental setups and best practices. In particular, I want to thank Philipp Schmidt and Falk Ebert for this reason.

Finally, I want to thank my family and my girlfriend, who have always been there for me. I am incredibly grateful for their support.

Erklärung

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Zur Überprüfung der Rechtschreibung dieser Arbeit wurden teilweise Grammarly und DeepL verwendet.

Heidelberg, den 14.06.2025,

Simon Tebeck