Photonics: Neuromorphic Optical Networks for AI

Nicholas Greig

ECE4094/ECE4095 Final Report

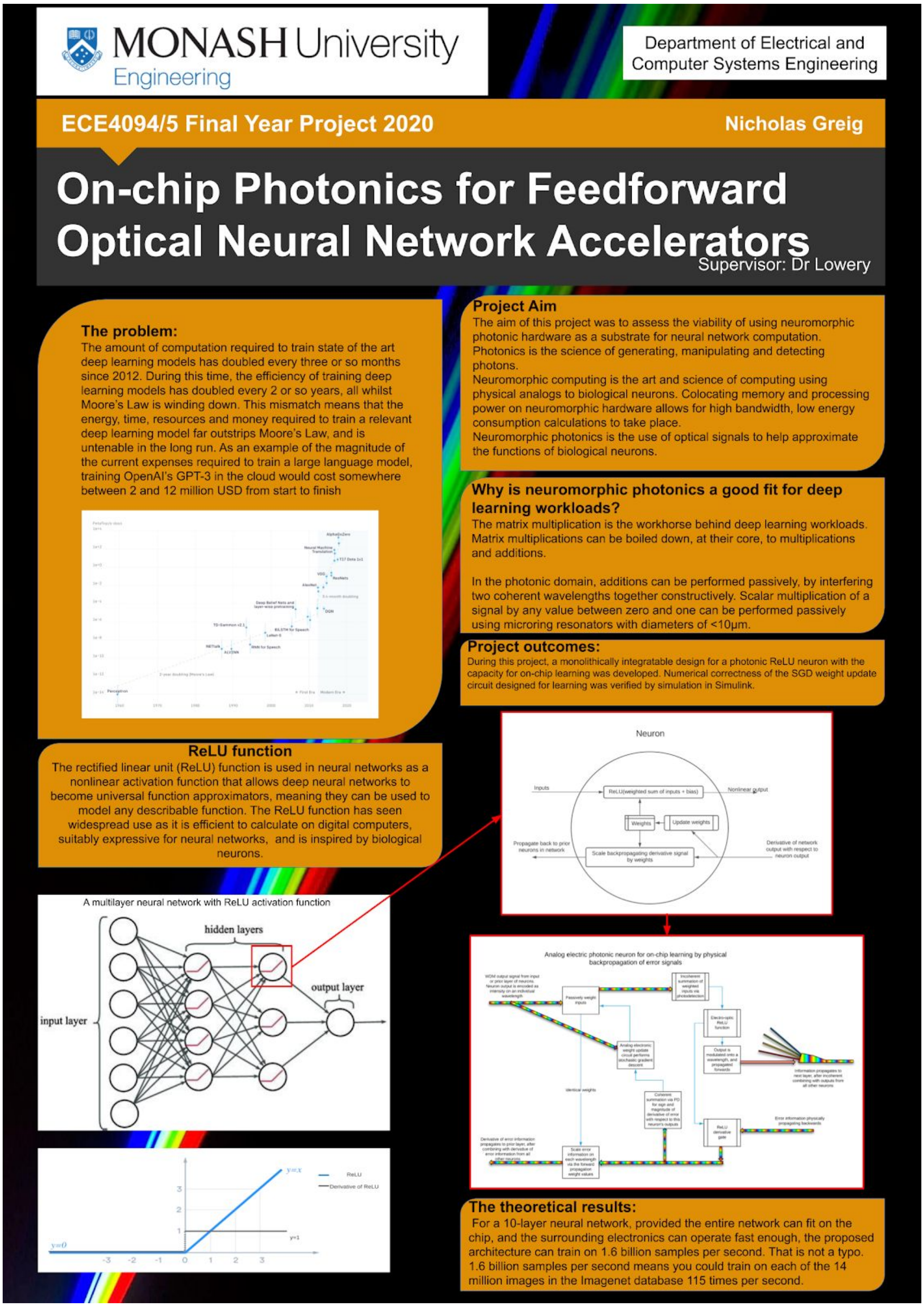
Department of Electrical and Computer Systems Engineering
Monash University

Date Submitted: November 6, 2020

Significant contributions page

An extensive literature review of the current state of photonic neural networks and photonic neuromorphic systems was created. A theoretic design for a photonic neuron that utilises physically backpropagating error signals and analog electronics to enable on-chip learning was developed. A simulation confirming the behaviour of the analog electronic circuits used to update the weights of the proposed photonic neuron was performed.

A copy of my poster



MONASH University Engineering

Department of Electrical and Computer Systems Engineering

ECE4094/5 Final Year Project 2020

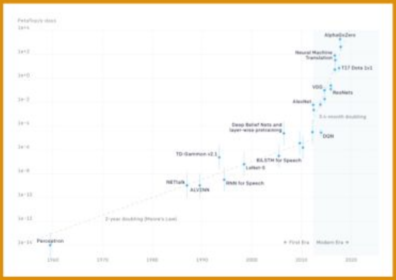
Nicholas Greig

On-chip Photonics for Feedforward Optical Neural Network Accelerators

Supervisor: Dr Lowery

The problem:

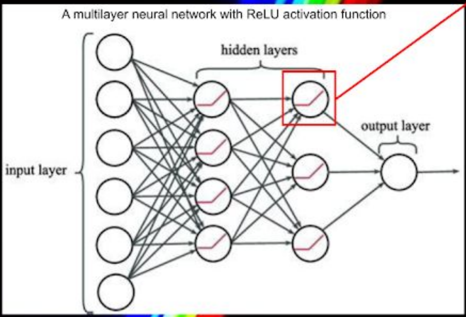
The amount of computation required to train state of the art deep learning models has doubled every three or so months since 2012. During this time, the efficiency of training deep learning models has doubled every 2 or so years, all whilst Moore's Law is winding down. This mismatch means that the energy, time, resources and money required to train a relevant deep learning model far outstrips Moore's Law, and is untenable in the long run. As an example of the magnitude of the current expenses required to train a large language model, training OpenAI's GPT-3 in the cloud would cost somewhere between 2 and 12 million USD from start to finish

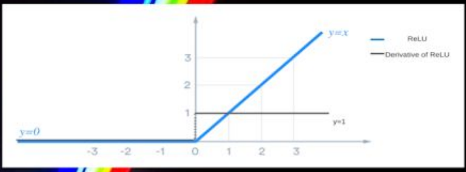


ReLU function

The rectified linear unit (ReLU) function is used in neural networks as a nonlinear activation function that allows deep neural networks to become universal function approximators, meaning they can be used to model any describable function. The ReLU function has seen widespread use as it is efficient to calculate on digital computers, suitably expressive for neural networks, and is inspired by biological neurons.

A multilayer neural network with ReLU activation function





Project Aim

The aim of this project was to assess the viability of using neuromorphic photonic hardware as a substrate for neural network computation.
Photonics is the science of generating, manipulating and detecting photons.
Neuromorphic computing is the art and science of computing using physical analogs to biological neurons. Colocating memory and processing power on neuromorphic hardware allows for high bandwidth, low energy consumption calculations to take place.
Neuromorphic photonics is the use of optical signals to help approximate the functions of biological neurons.

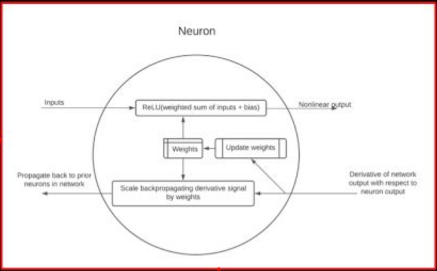
Why is neuromorphic photonics a good fit for deep learning workloads?

The matrix multiplication is the workhorse behind deep learning workloads. Matrix multiplications can be boiled down, at their core, to multiplications and additions.

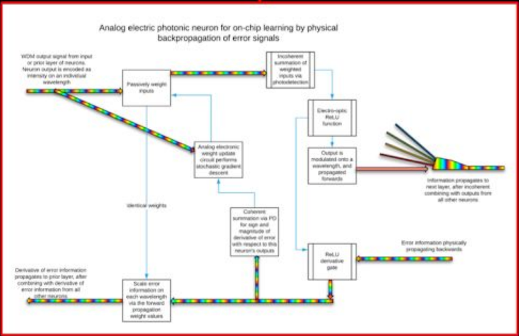
In the photonic domain, additions can be performed passively, by interfering two coherent wavelengths together constructively. Scalar multiplication of a signal by any value between zero and one can be performed passively using microring resonators with diameters of <10μm.

Project outcomes:

During this project, a monolithically integratable design for a photonic ReLU neuron with the capacity for on-chip learning was developed. Numerical correctness of the SGD weight update circuit designed for learning was verified by simulation in Simulink.



Neuron
Inputs
ReLU(weighted sum of inputs + bias)
Nonlinear output
Weights
Update weights
Propagate back to prior neurons in network
Scale backpropagating derivative signal by weights
Derivative of network output with respect to neuron output



Analog electric photonic neuron for on-chip learning by physical backpropagation of error signals
WDM output signal from input or prior layer of neurons. Neuron output is encoded as intensity on an individual wavelength
Passively weight inputs
Incoherent summation of weighted inputs via photodetection
Electro-optic ReLU function
Output is modulated onto a wavelength, and propagated forwards
Information propagates to next layer, after incoherent combining with outputs from all other neurons
Analog electronic weight update circuit performs stochastic gradient descent
Identical weights
Coherent summation via PD for sign and magnitude of derivative of error with respect to this neuron's outputs
ReLU derivative gate
Error information physically propagating backwards
Derivative of error information propagates to prior layer, after combining with derivative of error information from all other neurons
Scale error information on each wavelength via the forward propagation weight values

The theoretical results:

For a 10-layer neural network, provided the entire network can fit on the chip, and the surrounding electronics can operate fast enough, the proposed architecture can train on 1.6 billion samples per second. That is not a typo. 1.6 billion samples per second means you could train on each of the 14 million images in the Imagenet database 115 times per second.

Executive summary

The purpose of this report is to explain to the reader the current state of optical neural networks, with attention paid to photonic neural network architectures, both proposed and physically implemented, and their performance in terms of throughput, latency, energy consumption and relative on-chip area consumed. An additional goal of this document is to present to the reader an in-depth design proposal for a feedforward WDM photonic neuron capable of on-chip learning via physically back propagated error signals.

This project was approached as follows:

- First, a good understanding of the current state of artificial intelligence, neuromorphic systems, optical neural networks and photonic neural networks was developed, after reading a large number of research papers and reviews on the various topics.
- Secondly, an in-depth literature review of the field of photonic neural networks was constructed, paying attention in particular to on-chip photonic networks.
- Third, an abstract design for a WDM photonic neuron that can perform on-chip learning via physically back propagated error signals was created and subsequently iterated upon.
- Fourth, an analog electronic weight update circuit was designed and simulated in Simulink, to verify the possibility of calculating the SGD weight update rule in analog electronics.
- Fifth, a design document was constructed, and integrated into this report, which outlined the functional isomorphism between an abstract neuron in a neural network which uses the ReLU non-linear activation function and the proposed photonic neuron.

The main findings of this project are as follows:

- Given the incredible performances demonstrated by deep neural networks in recent years and the energy consumed and environmental impacts of training and running these neural networks is non-trivial, it is of tremendous importance that more efficient methods of computing these models are explored. The additional computational power provided by Moore’s Law year after year is outstripped by the increasing amount of computations required for deep learning workloads.

- The burgeoning field of photonic electric circuits for the computation of deep learning based workloads is a promising avenue of further research and innovation, as photonic integrated circuits can operate at low energy, high throughput and can be manufactured using existing, low cost manufacturing techniques.

- As a large amount of energy and time is consumed during the training process of a neural network, a photonic integrated circuit should be compatible with the training process of a neural network to allow for the most benefits of photonics to be adopted.

- The low size, weight and power constraints of edge computing make it an ill suited destination for photonic circuits, as photonic components are orders of magnitude larger than transistors, and are unlikely to shrink much due to the requirement of Server-side computing, wherein energy consumption and throughput are of greater importance than area constraints, is the ideal insertion point for industrial photonic chips that can accelerate deep learning workloads.

Acknowledgements

I would like to acknowledge the insightful, humorous and realistic supervision provided to me by Professor Arthur Lowery. Our meetings were always entertaining, reassuring and left me with a clear sense of what ought to be done next.

I would also like to acknowledge the help and domain expertise which Mike Xu lent me over the course of this project. His assistance in determining how best to compare and analyse photonic neural networks was greatly appreciated, as well as the various other insights he provided.

Table of Contents

Abbreviations

AI: Artificial intelligence
ANN: Artificial neural network
CNN:Convolutional neural network
CMOS: Conventional metal oxide semiconductor
CPU: Central processing unit
DRAM: Dynamic random access memory
GPU: Graphical processing unit
FINFET: Fin field effect transistor, which is a multigate MOSFET transistor
IC: Integrated circuit
LIF: Leaky integrate and fire
MOSFET: Metal oxide semiconductor field effect transistor
ONN: Optical neural network
PIC: Photonic integrated circuit
SNN: Spiking neural network
SNR: Signal to noise ratio
RNN: Recurrent neural network
ReLU: Rectified Linear Unit
SGD: Stochastic Gradient Descent, an iterative method of optimising a suitable differentiable function that is computationally simple.
STDP: Spike timing dependent plasticity, a behaviour of biological neurons whereby their potential to output a spike is dependent upon when their last spike occurred.
WDM: Wavelength division multiplexed, a term used to describe the coexistence of multiple, non-interfering wavelengths within a waveguide or channel

1. Project Introduction

This project is about the topic of neuromorphic computing, photonics and the growing intersection in both research and industry between the two fields. This project aimed to create a thorough literature review, introducing the reader to the basics of neuromorphic computing and photonics, and why neuromorphic computing, deep learning and AI workloads are ill-suited to conventional von Neumann architecture computers. Specifically, this project sought to demonstrate the potential benefits of constructing hardware accelerators for deep learning workloads which utilise integrated photonics components to transmit and transform information in both the electrical and optical domains. A proposal for a monolithically integrated photonic neuron that utilises analogue electronic circuits to perform a nonlinear activation function and weight updates is introduced, and stepped through, with a clear isomorphism to an abstract ReLU neuron a main focus of the design.

The potential benefits of utilising photonics as opposed to electronics to transmit and transform information are an increase in bandwidth per channel, increased switching speeds, decreased heat generation, reduced parasitic RC effects, and passive addition and scaling at essentially the speed of propagation through the medium. Recent advances in IC manufacturing have enabled the ability to manufacture chips with both electric and photonic components monolithically integrated, which reduces coupling losses, energy consumption and component size of PICs. Semiconductor foundries can cost billions of dollars to construct, so leveraging the capabilities of existing fabrication facilities is essential to reduce the barrier to entry of constructing PICs.

2. Photonic Neuromorphic Computing: An In-depth Review

Abstract

The deep-learning explosion has spawned renewed interest in photonic hardware implementations of neuromorphic systems, due to the energy inefficiencies and low-throughput of traditional von Neumann computer systems in simulating deep neural networks. The advantages of photonic encoding, transmission and transformation of information make it a desirable alternative to neuromorphic electrical systems. In this paper, the current state of AI is briefly summed up, and a summary and comparison of recent photonic neural network architectures is presented, with additional consideration given to physically implemented designs that accompanied a proposed architecture.

2.1 Introduction

When a person talks about AI today, they are almost certainly talking about a deep-learning model. Expert systems, coded based upon known rules and heuristics by champions and experts of narrow domains, were once considered the best chance at creating artificial intelligence. The number of rules and logic needed to be explicitly coded into such a system generally scales with the complexity of the system, and requires tremendous human input. As such, modern deep learning paradigms currently reign supreme in the world of AI.

For the case of deep reinforcement learning agents, they do not require explicit knowledge of the rules of the environment in which they reside, just a degree of freedom with which to explore their environment and some method to measure success, and they will figure out methods to maximise their success, albeit while requiring numerous (generally on the order of thousands) examples.

Supervised learners, such as image classification networks, do not require explicit knowledge about what makes a car different to a fox: just show them enough labelled examples of each, add some backpropagation of error signals to move the internal calculations in the direction that will reduce the global error of the system, and they will handle the rest. Generative deep neural networks can create photo-realistic human photographs[1], can relight portrait images[2] and perform near-realistic audio generation from text and a small sample clip[3], and even swap faces of individuals in videos[4].

In recent years, deep-learning models have reached unparalleled levels of accuracy on a number of important tasks such as cancer detection[5], scene recognition and understanding for automated driving[5], [6] and automated drug discovery[7]. Indeed, given that a deep neural network can approximate any function [8], it seems likely that all fields of scientific inquiry and industrial application will eventually come to leverage the capabilities of deep neural networks.

The major problem that exists with the (traditional) von Neumann architecture is that memory and computation are segregated, requiring both algorithms and data to be fetched from the memory to the CPU ahead of the execution of instructions. This was described as early as 1978, where it was coined as the “von Neumann bottleneck”[9]. While CPUs have grown faster in accordance with Moore’s Law, transfer speed and memory access lagged behind[10], making this bottleneck an increasingly important obstacle to overcome. Fetching data from disk memory can take tens of thousands of clock cycles, and even accessing smaller high-speed memory caches closer to the CPU still creates a bottleneck, with access times ranging from 3 to 50 clock cycles. Parasitic capacitance, as a result of the high speed switching of electrical states used for the shuffling of binary information, in wires limits both the increase of memory speeds and the reduction of energy consumption in modern computers. The von Neumann bottleneck is especially inconvenient in modern deep-learning models, as intermediate representations that can be orders of magnitude larger than the input data must be temporarily stored and then retrieved for later processing. In addition, the large parallelism of deep neural networks lends itself towards computation on dedicated hardware to increase the bandwidth and throughput of these networks.

The explosion post 2012 of deep learning was not due to better algorithms, although with renewed interest in the field several recent advancements have been made. A major algorithm behind deep learning, stochastic gradient descent (SGD) via backpropagation of errors[11], was first written about in the context of learning novel internal representations of information in 1986, but without large scale datasets and more computational power to train larger models, the field remained relatively stagnant. Experiments on small datasets could still be run, but the computational cost of training and running neural networks hindered industry adoption of deep neural networks. The utilisation of consumer-grade Graphical Processing Unit (GPU) cards to perform the basic functions of a neural network, namely forward propagation of information and back-propagation of errors, spiked renewed interest and the current wave of deep learning applications and research. By utilising the massive parallelisation of GPUs, training times, energy expenditure and model latency could be decreased by orders of magnitude, allowing for more experimentation at a lower cost and reduced latency in neural network inferences.

However, despite leveraging the increased computational power that came along with efficient GPU implementation of deep learning algorithms, the field is still quickly outstripping the speed at which GPU processing power can evolve, with an approximately doubling in the compute power required to train a state-of-the-art model every 3 months (Figure 1). This far outpaces Moore's Law (whether it is waxing or waning) and is simply untenable in the long run. With this increase in computation comes a noticeable increase in carbon footprint[12], and it also introduces latency into the software development process, as a result of the increased training times for larger, more compute-intensive models. Neither of these trends can be allowed to continue, as they impose greater and greater risks upon the environment and the continuous cycle of improvement.

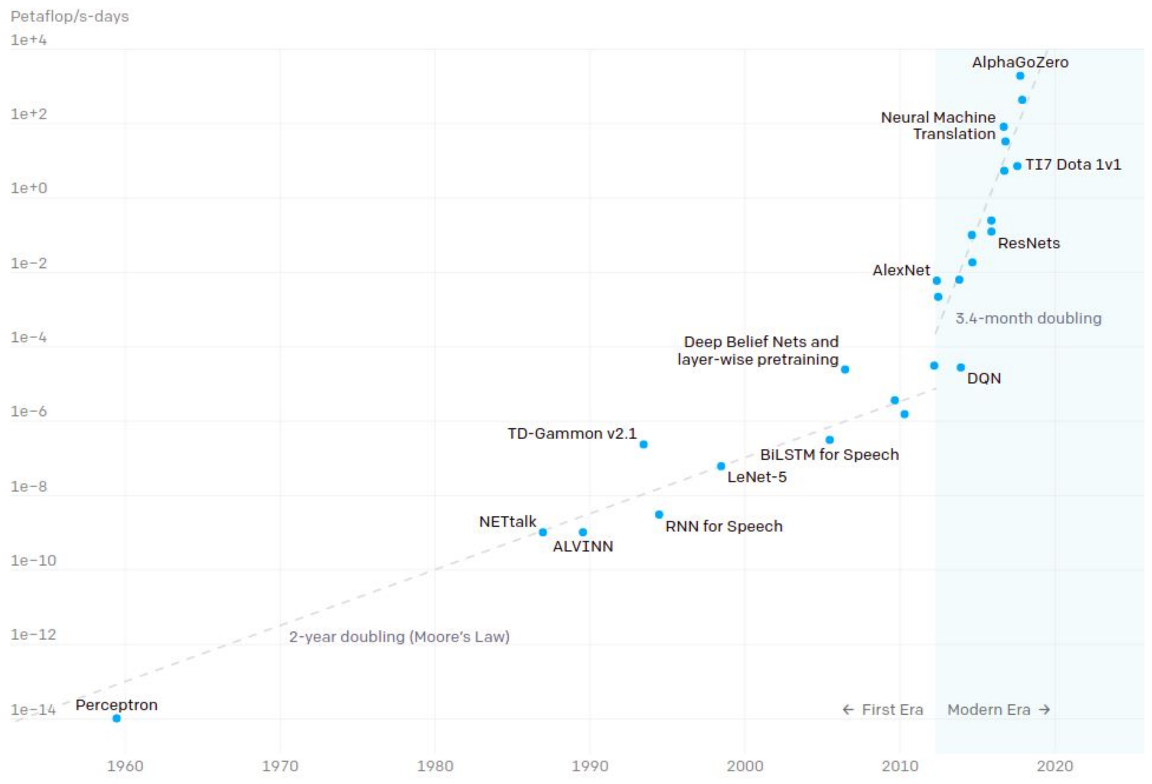


Figure 1: Compute required for AI research. A 300,000x increase in compute is seen from 2012 until 2018. Taken from[13]

2.2 Bypassing the Von Neumann bottleneck: Dedicated AI processors

A new generation of hardware accelerators is necessary: one which provides ultra low-latency for matrix-matrix multiplication (the kinds of calculation most often used in deep learning), high parallelism, low energy consumption and low on-chip spatial footprint. Existing AI hardware startups and technology industry giants, including Google, Tesla, Cerebras, Graphcore, Nervana, Habana, Intel, Nvidia, AMD, Wave Computing, Lightelligence, Luminous Computing and UntetherAI are all vying to create next-generation hardware accelerators for deep-learning.

The majority of efforts in this area focus on electrical implementations of non-von-Neumann architectures, however these are still constrained by parasitic RC effects, electronic interconnects which can only transmit a single bit per clock cycle, and electronics-scale global clock cycles (generally around 1-2Ghz).

The emerging field of photonic integrated circuits (PICs) is ripe for the deep learning task; matrix multiplications can be performed passively using optical components, at speeds orders of magnitudes greater than current CPU and GPU clock speeds. Interconnects can easily transmit data with Tb/s bandwidth by using multiple non-interfering frequencies within the same waveguide with negligible loss. Photonic components can often be manufactured using existing silicon foundry capabilities, reducing the need for entirely new manufacturing pipelines to be created and increasing the potential for monolithic integration of silicon photonics alongside traditional CMOS electronics.

One notable downside to photonic components is their size: they can be microns or larger in both length and width, in order to allow for interference, resonance and other desired optical effects to occur readily at microwave frequencies. In comparison to current 7 nm FINFET technology, which can create a single DRAM memory cell in as little as 0.027um^2 of on-chip space, photonic components are enormous; a small microring resonator, a typical photonic component for wavelength selection and amplitude modulation can be 5 micrometers or more in diameter [14]. However, given that photonic components can operate on multiple wavelengths simultaneously, and at rates orders of magnitude higher than current integrated electrical components, the tradeoff in component size may be worth the decreased component density. Additionally, PIC sizes may be shrunk with better component placement, advances in component design and other material and manufacturing discoveries and variations. The passive, low-loss nature of photonic components means that 3D designs can be effectively realised without excessive concern for internal thermal regulation, a key limiting factor in 3D integrated electronic circuits.

Another caveat of utilising photonic computing opposed to electrical computing: to reduce the number of waveguides and components used, computation is usually done using analogue signals, which must then be converted back into the digital domain. There is an inherent robustness to clock-based computational paradigms, as engineers can set the clock rate as lesser than the settling time of the logic gates, ensuring that high resolution, zero error rate calculations can be performed with a sufficient number of logic gates. This is not the case in the analogue domain, as resolution of calculations is limited by the highest SNR achievable in the system. However, analogue electronic training and inference of deep-learning models has already been displayed by the Analog AI team at IBM, with experiments showing only a minor degradation in accuracy from computing in the analog domain in comparison to 32-bit digital calculations[15].

This result aligns with recent works in the deep learning world, showing that full-precision (32 or 64-bit weight and activation) networks are not necessary; with well designed training protocols and networks, only a 2% drop in accuracy is reported by switching to 1 bit width weights and 1 bit activations[16]. Since analogue control of photonic signals with 5 bit resolution has been demonstrated[17], computational precision in the analog photonic domain should not be a limiting issue in the performance of neural networks accelerated by processing in the photonic domain, and the available throughput and energy efficiency advantages should outweigh this constraint. Indeed, some inherent noise may even help the processing capability of neural systems[18][19], by acting as a regularizing force, helping the network to avoid local minima.

Alternatively, digital computations using photonic components [20] are possible, but the size and number of photonic components required for digital calculations at high throughput present a notable barrier to entry.

The remainder of this paper is structured as follows: we will outline the purported benefits of neuromorphic computation and the current state of electrical neuromorphic chips, briefly discuss the different categories of neural networks, introduce (to unfamiliar readers) the potential benefits with respect to electronic computation that photonics can bestow, and lastly an in-depth summary and comparison of recent Optical Neural Networks (ONNs) is given.

2.3 Neuromorphic computation

The way in which brains calculate sums and arrive at decisions is performed on an entirely different substrate and layout than that of modern computers. Neuromorphic computation is the term given to mimicking the behaviour of biological neurons, the simplest atomic component in a biological brain, in artificial hardware.

Biological neurons exhibit a host of complex functions, and have fan-in (the number of input neurons each neuron is connected to) and fan-out (the number of neurons each neuron propagates an output to) on the order of thousands or tens of thousands. Connections between neurons are called synapses, and after the firing of a neuron, synapses can exhibit altered behaviour on time scales from milliseconds to hours long.

Accurate abstractions and computer models of biological neurons do exist, but are prohibitively expensive to run en masse on von Neumann computers. For example, a single neuron based on the Hodgkin Huxley model[21], which requires solving a set of nonlinear differential equation, involves more than 1000 calculations to simulate 1ms of neuron activity, and this is simply not scalable to larger networks of neurons on current von Neumann hardware[22].
Simpler models, which retain select features at the expense of omitting others, such as the leaky integrate-and-fire (LIF) model[23], have been proposed, but there remain unanswered questions as to which functions contribute most to the intelligence exhibited by biological brains. Determination of the functions which are essential to information processing, and which are vestigial byproducts of both the evolutionary processes which developed biological brains and the state of the cell as a living being itself, remains an open problem.
An intermediate solution, the Izhikevich neuron model[22], [24] achieves high biological plausibility with relatively low computational complexity. Izhikevich neurons are often considered when trying to match observed neuron activity to a model, but have seen less

use in designing neural networks from scratch than the LIF model, perhaps due to a lack of specific learning rules.

More complex, unsupervised interneuron activities, such as Spike Timing Dependent Plasticity (STDP) [25]–[30] have been extensively studied after experimental discovery and verification in the 1990s. STDP is a behaviour exhibited between neurons wherein a neuron will strengthen connections with the neurons that provide it relevant spike information (spikes likely to lead to it emitting a spike) and weaken connections with neurons that propagate a spike to it after it has emitted a spike. STDP is a biologically plausible form of Hebbian learning[31], the source of the phrase “Neurons that fire together, wire together”, although Hebb did not posit that synapse connections could be weakened as a result of the firing of neurons. Biologically plausible inter-neuron connections are an additional function that increases the complexity and memory required of simulations of biologically accurate networks of neurons on von Neumann architectures and increase the latency of such simulations, but can be implemented relatively easily using controllable, dedicated analogue electronics in a neuromorphic architecture.

Neuromorphic architectures are more useful the more neuronal functions they can exhibit and be reconfigured to perform, but the hardware implementation of additional functions comes at the cost of decreased neuronal density, increased power consumption, more complex inter-neuron communication regimes and more expensive ICs. Hardware which operates specifically on the creation and processing of spikes is doubly useful: it can be used to both study models of brain function and run energy-efficient deep neural networks. Some neuromorphic hardware doesn’t utilise very specialised dedicated electrical hardware to perform neuron functions: the SpiNNaker project[32] uses multiple parallel ARM microprocessors which are fully reconfigurable to perform any computation. This comes at the cost of energy efficiency: in Table 1 below, it can be seen that SpiNNaker uses 1-4 orders of magnitude more energy per synapse operation than other neuromorphic architectures.

Still, simple neuromorphic hardware exhibits many attributes that are desirable for the modern world. The reduction or elimination of the von Neumann bottleneck is one such benefit, and potential decreased complexity of hardware design (due to the removal of global clock signals), reduced energy usage and inherent fault tolerance are other potential positives of this pursuit. The ability to perform on-chip training of neural networks on neuromorphic hardware is an additional goal of the neuromorphic design community, as it would lead to a reduction in the energy used and time required to train neural network models and would streamline the deployment process of trained networks to neuromorphic accelerators.

By removing the distance between memory and computation, global clock signals can be eliminated, and as a result components (such as neurons) can remain idle until some input event triggers their activity, which can drastically reduce energy usage without explicit energy management schemes.

The purported benefits of purely neuromorphic systems come with a tremendous caveat: they require a substantial shift away from the past 70 years of software and hardware design, and towards new methods of program verification, systems design and engineering.

Engineers may need to forego the strict design and modelling of systems in favour of understanding the broader mechanics of self-organizing sub-networks realised in neuromorphic systems. The merits of the existing system of von Neumann computers are too large to be tossed aside, and so the future will likely contain a merging of von Neumann computers and neuromorphic architectures into hybrid computers, with tasks delegate to one or another paradigm depending upon the circumstances.

Table 1: Existing electronic neuromorphic chips and their characteristics (adjusted from [33])

Processor	BrainScaleS [34]	Neurogrid[35]	TrueNorth [36]	SpiNNaker [32]	Loihi[37]
Implementation	Analog	Analog	Digital	Digital	Digital
Time	Discretized	Real or continuous time	Discretized	Discretized	Discretized
Neuron update	Time MUX	Real time	Time MUX	Time MUX	Time MUX
Synapse resolution (in bits)	4	13, shared	1	Variable	1-64
Biomimicry	Not configurable	Not configurable	Limited to LIF	Configurable	Configurable
On-chip learning	STDP only	No	No	Yes	Yes
Neurons per core	8-512	65,000	256	~1000	1-1024
Synapses per core	~130,000	100,000,000	6,500	1,000.000	16,000 at 64b
Cores per chip	352 (wafer scale)	1	4096	16	128
Chip area (mm^2)	50 (single core)	168	430	102	60
Technology (nm)	180	180	28	130	14 (FinFET)
Energy/Synapse Operation (pj)	174[q]	941[a]	27[b]	27,000[c]	Minimum 105.3[d]

[a]calculated as: total power/(spike per second X number of synapse), data from [35]
[b]calculated as: total power/(number of neurons × firing frequency × number of synapses), data from [36]
[c]calculated as: total power/number of connections per second, data from [32]
[d]calculated as: active neural update + synaptic operation + within tile transmission, data from[37]

2.4 Neural networks

As early as 2012, artificial feedforward neural networks had already achieved superhuman levels and levels higher than handcrafted programs in a number of specific, narrow pattern recognition and visual classification tasks [38],[39].

The traditional artificial neural network (ANN) is based upon a sequential network of parallel “neurons”. Each “neuron”, loosely based upon the neurons in a biological brain, can transmit a signal to other neurons, depending upon its current ‘state’ and the inputs it receives. Generally, a neuron will multiply each input by some linear weight, add a bias and then apply a non-linear activation function.
The output of each neuron **y** is described in Equation (**1**), where **f()** is a non-linear activation function, **x** is a vector of n inputs, **b** is a vector of biases and **w** is a vector of weights.

$$y = f\left(\sum_{i=0}^{n} w_i \cdot x_i + b_i\right) \tag{1}$$

In both feedforward and recurrent neural networks, the weights and biases of a particular neuron are stationary, and do not change while performing a forward pass over the network. During training, backpropagation of errors are used to adjust the values of these weights and biases, in order to minimise some global optimisation or loss function.

With each neuron receiving input information, storing its own weights and biases, and performing computation, the efficient operation of an ANN on von Neumann computer architectures requires complicated memory management and optimisation, in order to reduce the amount of memory fetches that occur whilst the network’s forward pass is occurring to reduce the latency and increase the throughput of the network.

2.4.1 Feedforward ANN

In a feedforward ANN, there exist no recurrent connections between neurons. No neuron will receive or act upon a signal or transformed signal which it has already fed into the network. Such feedforward networks can be considered as Directed Acyclic Graphs (DAGs), in that for each single input fed into the network, also known as a forward ‘pass’ of the network, each neuron will only compute and transmit a signal once.

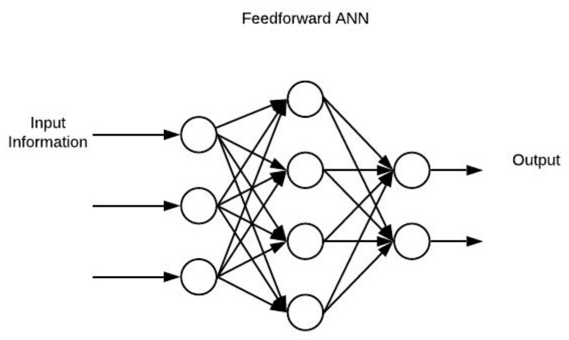


Figure 2: Simple Feedforward ANN

Convolutional Neural Networks (CNNs)

A CNN can be considered a subclass of feedforward ANNs, as each ‘neuron’ in a layer is connected to at most *k* ‘neurons’ in previous layers, where *k* is the size of the receptive field kernel. For example, for a two dimensional 3x3 kernel which connects layer *i-1* to layer *i*, each neuron in layer *i* is connected by a set of shared weights (the kernel *k*) to a spatially constrained set of neurons in the layer *i-1*, such that each neuron in *i* shares the same weight kernel *k*, and the number of neurons which *k* samples is less than the number of neurons in layer *i-1*. As such, CNNs are subsets of ANNs with limited receptive fields, and shared, repetitive weights for each layer of neurons. CNNs are used in various natural-image processing tasks, and hold the state of the art on numerous challenges wherein spatially-related information is available.

2.4.2 Recurrent ANNs (RNNs)

In a recurrent neural network, the outputs of neurons later in the network can be fed to earlier neurons, forming cyclic loops in the graph. This can allow for multiple inputs to be fed into the network in a sequence, with each input changing the output of later inputs, and as such is particularly useful for modelling or otherwise classifying data which is inherently sequential, such as human language and time-series information. Importantly, RNNs as they are commonly used today are not continuous with respect to time: neurons only output information or change internal state when a new signal is fed to them. Also, RNNs are continuous with respect to values, meaning that for each new input fed into the network, each neuron will propagate information forward. Even if the result of the neuron’s activation function is a 0, it still sends this 0 forward to all the neurons it is connected to.

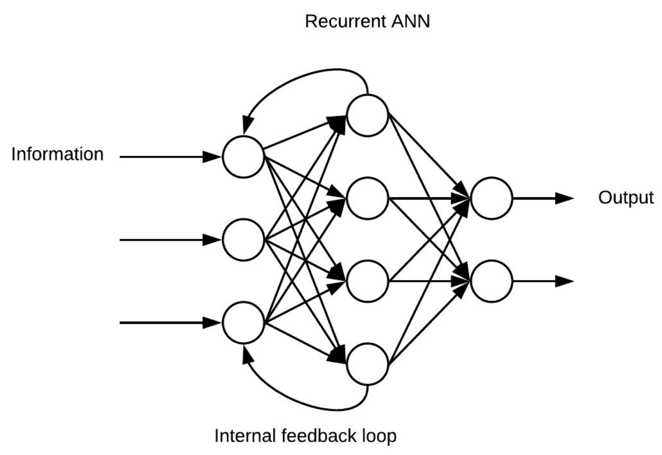


Figure 3: Simple Recurrent ANN

Until 2017, Recurrent ANNs and Long Short Term Memory (LSTM) networks, a variant of RNNs with more complex recurrent neuron update rules, held the state of the art in various speech and language based translational tasks, question answering, and reasoning tasks.

They have fallen out of favour on these tasks recently, as the Transformer [40] network architecture surpassed RNNs on a number of public benchmark datasets relating to natural language, and is easier to train and less computationally expensive. A Transformer network is not strictly an RNN, although during inference for translational tasks it requires recurrently feeding intermediate outputs back into itself as inputs.

Recently, architectures such as Turing-NLG[41], a transformer-based language model recently released by Microsoft’s deep learning research team, which has 17 billion parameters (each stored in 32bits), requires 8 GPUs to store the model parameters and trains with a batch size of 512 using 256 GPUs connected in parallel, have begun to dominate the subfield of deep learning based natural language processing.

Recurrent neural networks still hold state of the art records for performance in domains regarding time-series information that is not language based, such as action prediction in video, time-series anomaly detection and stock price fluctuation prediction, to name a few.

The computational complexity and memory intensity of RNNs with long memory time-spans presents a bottleneck to the expansion of these networks, as intermediate values for each time-step must be held in memory to then perform back-propagation through time, and the rate of model growth far outpaces the growth of on-chip memory available. Much work is done by software engineers on parallelising these networks, such that GPU clusters and locally available memory can actually fit these models and train them.

As an aside, creating models this size concentrates research capabilities and applications into the hands of only the richest, most resource-laden universities and corporations. It also bodes poorly for the energy consumption and computational efficiency of the future of AI research.

2.4.3 Spiking neural networks (SNNs)

Spiking neural networks are ANNs that mimic truly biological functions closer than their other ANN counterparts. With inputs coming into the network as continuous time-domain ‘spikes’ of information, spiking neural networks are more similar to the way our brains continuously function than RNNs or ANNs. Convolutional SNNs [42], [43] have been recently proposed and experimentally verified, and in some cases exhibit almost parity performance with traditional CNNs.

Some spiking neural network implementations do not use continuous time to update the internal state of neurons, but instead employ an event based update system, where neuron updates are only calculated when it receives an input spike[44], [45]. Such event-based signalling, known as Address Event Representation (AER) in asynchronous systems is employed by Intel’s Loihi[46] neuromorphic integrated chip, but this is a noticeable divergence from biological isomorphism with respect to [47]–[50]continuous-time updating of neuron states.
AER allows small area consumption and low operating power chip designs, but each packet of information transmitted includes information about the relative or absolute timing of the event or spike, as well as the address of the neuron that created it and the neuron it is being sent to. This additional information about global or relative timing generally requires that the chip incorporate a timing signal either locally on each neuron or globally to each neuron, and as such poses multiple additional requirements on the design.

Information in a spiking neural network can be encoded in a number of fashions: rate coding[51], [52] and temporal coding are two of the major schools of thought.
Broadly speaking, rate-coding involves transmission of data as the mean-firing frequency of a neuron, where larger frequencies mean larger values.
Temporal coding involves encoding information into the exact firing time of spikes, and a subset of this regime is the coding of importance of spikes into the latency of the spike, wherein the most important information is propagated first and the strongest activated neurons fire the earliest[47], [48].
There is evidence to suggest that information in the human brain cannot be processed using rate coding; biological neurons can fire about every 100ms, yet humans can classify images in under 150ms, which is too short a time to create a mean value of the spiking state of a neuron[53]. Recent work in temporally coded network backpropagation algorithms[54] has enabled new benchmarks on common image classification datasets such as MNIST[55]. The idea that only a single spike from each neuron transmits sufficient data for classification of complex data has been experimentally verified in [54], [56], [57].

At the time or writing, spiking neural networks have not surpassed the easier to implement ANNs on a variety of baselines, despite research which shows that spiking neural networks can compute everything that a more traditional ANN can[58], but they are closing the gap substantially.
There are multiple potential reasons for this, which we shall briefly discuss:

- Discretization of time and Computational cost
- Lack of high-performance algorithms
- Relative difficulty of simulation
- Relative lack of research interest

The author notes that this list is not exhaustive, and more reasons may exist.

Discretization of time and computational cost

There are inherent difficulties in simulating a SNN on von Neumann architectures; since SNNs are continuous with respect to the time-domain, and von Neumann architectures operate on discrete clock-cycles, there is an inherent lowest time resolution that is simulated with an SNN, and this is set by a software/hardware engineer.
Whilst the clock-rates of an SNN in hardware can be orders of magnitude faster than biological neuron spike rates (human brain cells operate at around 100Hz at most), neurons can emit a spike of information an any point in time, whilst a discrete-time simulated SNN can only emit a spike at discrete time-steps. The designer can seek to reduce error associated with the discretisation of time by decreasing the size of the time steps being simulated, but this comes at the cost of requiring more time per neuron to simulate its activity faithfully.
Since spike information in a true SNN is continuous in time throughout the entire network, the output is also continuous in time, and so each forward pass of the network requires continuous, non-trivial time to complete.

Further, since the training time of SNNs is substantially larger than ANNs, they are often not trained directly. Instead, an ANN is trained as normal, and then the weights and architecture are converted to a SNN. This may lead to noticeably different results, as the conversion from a time-discrete analogue valued artificial neural network to a time-continuous spike-based neural network may introduce errors. In addition, networks designed for use with single-time-step input information that are converted to SNNs may not fully utilise the computational richness available with the added time dimensionality inherent in SNNs, and as such SNNs could stand to benefit from domain-specific network design, made more feasible with domain-specific hardware.

Lack of high performance algorithms

A lack of algorithms to train SNNs may also be a contributing factor to their reduced performance with respect to continuous-valued ANNs. Generally, SNNs are made by either converting a backpropagation trained ANN model to a SNN[59], [60], using continuous gradient approximations or other transformations of spikes to allow training by backpropagation[51], [52], [61]–[63], or by using more biologically inspired functions to train the network, such as STDP[64].

Relative lack of research interest

One last reason that SNNs may not yet be competitive with their discrete-time analogues is a simple lack of research interest. Whilst the neuromorphic engineering research field has existed since Carver Mead first coined the term in the 1990[65], the amount of relative interest in the low-hanging fruit exhibited by ANNs is orders of magnitude greater than the resources invested in neuromorphic systems and spiking neural networks, leading to a mismatch in the progress of each respective sub-field.

2.4.4 Summary of differences between ANNs and SNNs

Table 2: Differences between ANNs, CNNs, RNNs, SNNs and biological brains

Network Attributes	Feedforward ANN	CNN	RNN	SNN	Biological Neural Network (human brain)
Fan-in	Commonly 10^2-10^3	10^3-10^5(generally 1x1, 3x3, 5x5 and 7x7 spatial kernels are used on 100s or 1000s of channels)	Similar to ANN. Large, can be on the order of 1000s	Large, on the order of 1000s (same as ANN).	10^3-10^4
Fan-out	Commonly 10^2-10^3	Commonly 10^2-10^3	Commonly 10^2-10^3	Commonly 10^2-10^3	10^3-10^4
Neuron states (such as weights of interconnects) vary over time while inferring	No	No	Only time-sensitive portions of the network, such as LSTM cells. No internal state change without change in input.	Yes.Time-domain coding of inputs and intermediate spikes is an important feature of spiking neural networks (either rate-coding or temporal coding). Neuron states can change between inputs, and may depend on the relative time	Yes. STDP and other synapse depression or excitationary effects can last for seconds, minutes, or even hours.

				since firing of a signal	
Common learning regime(s)	Gradient descent (GD) by backpropag-ation of errors	GD by backpropa-gation of errors	GD by backpropa-gation of errors across time-unrolle d network	GD by approximate backpropag-ation of errors, STDP	STDP, unknown

Whether using an SNN or a traditional ANN, literature has shown that the resolution of the incoming information is not as important as the fan-in and fan-out: works in binarized neural networks[16][66][67], have shown commendable performance relative to full-precision networks utilising 4 bits of information or less. As such, photonic neuromorphic computers designed for AI based workloads may benefit more from reduced input bit resolutions, and higher fan-in and fan-out.

2.5 A brief introduction to silicon photonics

Photonic components have a number of advantages over electrical components. Due to the lack of interference between different wavelengths of light, and the low-signal degradation of silicon waveguides (as low as 0.045dB per metre in SiN[68]), multiple different streams of information can be encoded in the same waveguide with minimal signal loss.

The atomic operations of any linear transformation, multiplication and addition, can both be performed passively with negligible energy loss in photonics[69]. Addition of two coherent waveforms is as simple as coupling the two waveforms into one, where constructive interference will occur and the output will be a sum of the inputs.

Scalar multiplication between zero and one can be performed in the arms of a Mach Zehnder Interferometer(MZI), wherein a single light source is split in two, one travels slightly further than the other, or slower than the other in the case of an electrically modulated MZI, and then they are recombined. Essentially, this corresponds to interference of two phase-shifted, identical wavelengths, and the resulting wavelength can be thought of as the original wavelength multiplied by the cosine of phi, where phi is the phase shift between the respective waveforms.

Scalar multiplication between zero and one of a wavelength's intensity can also be performed via the intensity reduction of light coupled across a tunable Micro Ring Resonator (MRR), so arbitrary matrix multiplication can also be realised by a reconfigurable, interconnected series of MRRs, optical amplifiers, combiners and splitters.

With active amplifiers like Semiconductor Optical Amplifiers (SOAs), Erbium-Doped Fiber Amplifiers (EDFAs) and LASERs, multiplication of optical signals by values greater than 1 can be performed, and these components can be manufactured to work on picosecond timescales.

Given the above, arbitrary matrix multiplication can be implemented using only a small number of active components and passive optical transformations. With recent advances in integrated optical amplifiers, these components consume very little power, on the order of microwatts or less, at speeds orders of magnitude higher than traditional electronics.

2.5.1 Interfacing photonics with electronics

In order to interface analog photonic components with existing digital computer architectures, we require methods of converting electrical signals into light, and conversely converting light into electrical signals.

Electro-optic conversions

Converting electrical signals into optically encoded information can be done by modulating the intensity of light via an electrically tunable filter. The thermo-optic effect describes the change in the refractive index of a material with respect to a change in the temperature of the material. The electro-optic effect, which utilises electric fields and material responses, can also be employed to modulate incoming light via changing the refractive index of a medium. Other common electro-optic conversions involve directly converting electrical information into light, using components such as lasers and LEDs.

Opto-electric conversions

The most common form of opto-electronic conversion is via photodetector. Converting information from the optical domain to the electrical domain via photodetector hinges heavily on two factors: the wavelength of incident light must be great enough to strip electrons from their parent atoms, and the amount of current generated is proportional to the intensity of incident light. The voltage created is fundamentally limited by the bandgap energy of the semiconductor photodetector, and there is an inherent trade-off between the desired current and the desired voltage produced by light incident upon a photodetector. Adjusting for more or less current, and thus less or more voltage, can be done by altering the resistance of the circuit as seen by the photodetector, or changing the composition of the photodetector semiconductor.

Other, more exotic forms of opto-electronic conversion, involving optical phase-change materials exist, but are more difficult to integrate and require complex manufacturing schemes. In addition, phase change materials (PCMs) such as chalcogenide ($Ge_2Sb_2Te_5$) currently operate on timescales and power requirements substantially larger than other photonic components (nanoseconds and mW respectively), somewhat mitigating the desired benefits of a photonic neuromorphic system over an electric neuromorphic system.
The above methods only convert information in the optical domain into electrical information, but in order for this information to again be useful in modern computing, it generally has to be converted to a digital signal, by an analog-to-digital converter (ADC). Integrated ADCs that can operate in the GHz domain with 16 bits and higher resolutions on 4 or more channels are available for commercial purchase, so achieving optical domain throughputs

only requires an array of ADCs equal to the size of the output of the system being developed.

2.6 Information in photonic systems

In order to fully utilise the broad bandwidth of photonics, system-level architecture must be designed with the encoding and decoding of information held at the forefront of the design process.
Information in the photonic domain can be encoded in a variety of different dimensions, including but not limited to the intensity, wavelength, polarisation and mode of a signal.
Here we briefly touch upon different methods of encoding light. This section is by no means exhaustive, but hopefully serves as an informative introduction into optical encoding for novice readers. Importantly, more involved methods of encoding optical information, and analogue information in general, such as phase shift keying, non-return to zero coding, return to zero coding and other more advanced forms of encoding information will not be considered; instead, the orthogonal dimensions onto which information in the optical domain can be encoded shall be discussed.
In order for a ‘dimension’ into which information can be encoded, transformed and decoded to be useful, some measure of control and understanding of the behaviour of that dimension must exist. Additionally, the ability to detect changes to that information dimension easily is required.

Intensity

The intensity of light, or the amplitude of the transverse electromagnetic waves which make up light, is the simplest way in which to encode light. Light intensity can be reduced using destructive interference, or increased using coherent constructive interference from two different sources or waveguides. The intensity of light can be converted into electrical current in the electrical domain using a photodetector, and electrical signals can be converted into intensity information using electro-optic modulators or active light generators such as lasers and LEDs.

Wavelength

Information on different wavelengths or frequencies of light can be scaled independently, for example by using wavelength specific MRRs. The intensity of light on different wavelengths can be measured by coupling light across wavelength specific filters, such as MRRs, before photodetection. Incoherent summation of the intensity of light on different wavelengths can also be performed by routing photons from different wavelengths onto a photodetector.

It is possible to convert light from one wavelength onto another, via opto-electronic conversion to current before conversion back into optical signal. Since the majority of optical components have resonances at varying wavelengths, in PICs engineers are not free to convert any wavelength into any other wavelength, but are instead limited to working with discrete wavelengths. As such, the number of wavelengths utilisable is discrete, but can be on the order of tens [70] or higher simultaneously propagating in the same waveguide.

Polarisation

Since photons of the same wavelength but different polarisations do not interfere, either destructively or constructively, creating a PIC with waveguides that carry multiple polarisations of light is a potential future method to increase the fan-in and fan-out of photonic neurons. Recent work has demonstrated polarisation-selective MRRs[71] are possible with passive MRRs of specific geometric dimensions, such that using at least 2 polarisations of the same wavelength of light could be used to effectively transmit information, essentially boosting the number of available channels in a waveguide, albeit at the cost of an increased complexity of manufacturing.

Timing

Information can also be encoded into the timing (relative or absolute) of spikes or large changes in intensity of light. Relatively close timing of spikes between separate neurons on different wavelengths may represent values that correlate highly. Latency coding, wherein the magnitude of a value is based upon the relative time of the spike, and the lower the latency the larger the proposed magnitude, is also a viable option for information encoding in optical neuromorphic systems.

Encoding information into the relative timing of spikes is of particular neuromorphic appeal, since it is analogous to the winner-take-all signalling often viewed in the brain. Whilst a CW approach to signal transformation would utilise individual time-slots onto which information can be encoded, spike-timing based approaches assume continuous time, and the relative time between signals can be used to encode the relative intensity or importance of each signal.

2.7 Summary of photonic neural network architectures proposed by recent research

Recent advances in silicon photonics that leverage the strengths of the CMOS wafer fabrication industries, as well as renewed interest in the applications of deep learning, have led to renewed interest in implementing neural networks using photonic components.

An implementation of a neuromorphic photonic network can be placed into one of four categories: reservoir computing, fully-connected neural networks, convolutional neural networks and spiking neural networks.

Of the four categories above, convolutional neural networks are a subset of fully-connected neural networks, but can be categorised alone, as the repeated use of a single kernel or set of weights for each neuron in a layer can lead to some useful parallelization, repetition of the same neuron weights, and data pipelining. Additionally, fully-connected neural networks are different from reservoir computing and spiking neural networks, in that there is no explicit internal connection change without backpropagation in a fully-connected neural network.

Reservoir computing[72]–[74] involves the use of a dynamic, non-linear spatio-temporal set of fixed processes that output nonlinear signals, which can be linearly weighted to perform useful computation. Only select portions of a reservoir network are trainable and changeable, as the nonlinear reservoir in which the generation of multiple different nonlinear permutations of inputs is created is implemented as a black-box of sorts; it is a physical system with complex time-dependent dynamics. Special care must be taken when employing reservoir computing to use a reservoir substantially complex to create a rich environment of nonlinear signals from inputs, without being so complex as to unnecessarily negatively influence the throughput of the system, by restricting the latency at which inputs can be fed into the reservoir. Additionally, larger reservoirs generally require more input power to effectively stimulate enough of the reservoir to perform sufficiently rich non-linear transformation of the input signal, which decreases the potential power and throughput benefits of photonic neuromorphic systems over electric neuromorphic systems.

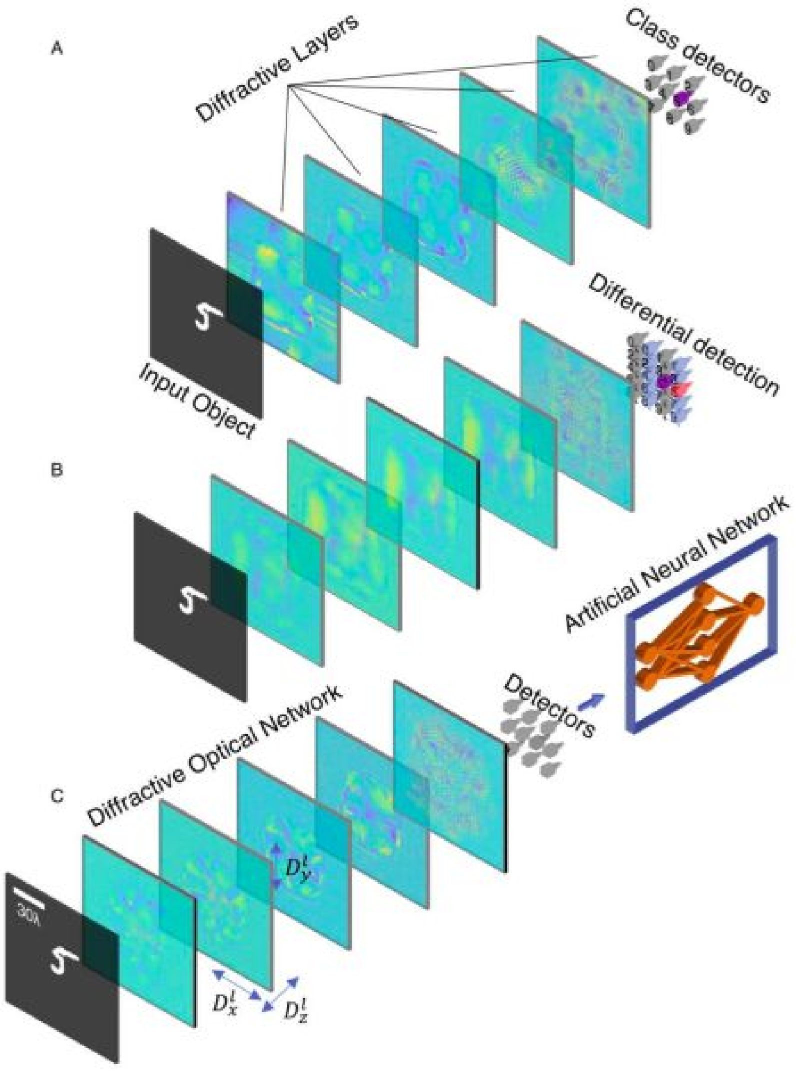


Figure 4: Different types of deep diffractive neural network based image classification systems. Taken from [75]

Optical neural network architectures which rely on diffraction, fourier-space transforms and free-space optics [76], [77][78][79][80] have been proposed and researched in depth, but the enormous size, inflexibility and sensitivity to alignment of such networks limits their industrial applicability. Efforts to reduce the alignment sensitivity of these free space networks was performed in [75], but the large size and relative inflexibility of these approaches still render them currently infeasible for real-world applications. The diffractive screens are generally created by additive manufacturing once a network has been simulated and trained on a digital computer, and this means that deep diffractive neural networks are often inference only, and cannot easily be adjusted and reprogrammed for new tasks without additional manufacturing and alignment steps, requiring human intervention.

In [81], densely manufactured three dimensional photonic interconnects are proposed and verified to increase the number of IO-channels available per unit area to connect to a neural network. Moughames et al. demonstrate that weighted interconnects which can implement specific convolutional filters, such as Haar filters, are possible using these 3D photonic circuits, and they posit that such photonic routing could prolong the existence of Dennard scaling in von Neumann computers, by allowing for densely packed information interconnects between traditional electrical logic gates. The essentially lossless propagation of light in waveguides allows for 3D interconnects to operate effectively without requiring advanced thermal regulation techniques, but still requires on-chip creation or modulation of light, which inevitably increases manufacturing complexity and costs of digital architectures.

Some PIC approaches to create ONNs envisage all-optical nonlinearities[82][83], whilst others involve opto-electrical conversions to implement a non-linear activation function[84][85]. Utilising purely optical nonlinearities generally allows for larger throughputs, but at the cost of increased power consumption, since optical nonlinearities such as saturable absorption absorb light otherwise used for signal propagation. The non-linear all-optical activation mechanism proposed by Miscuglio et al. has not yet been physically realised, but was simulated extensively.

Optical STDP realised via electro-optic modulator and SOA was demonstrated in [86][87], and an experimental demonstration of STDP based learning of spike-times was performed in [88], demonstrating controllable power and time response characteristics. Tunable power optical STDP is numerically simulated in [89], by utilising a VCSOA attached to an optical circulator. Passive coupled second-order MRR behaviour is simulated in [90] to perform optical STDP, based on TPA and an increase in the availability of charge carriers leading to a frequency shift.

A number of proposed PIC ONN architectures [83], [91],[84] utilise coherent light sources and MZIs to modulate incoming light intensity and transmit information, while others[[70], [92],[93] employ WDM signals to transmit information.

2.7.1 Temporal encoding and second order dispersion

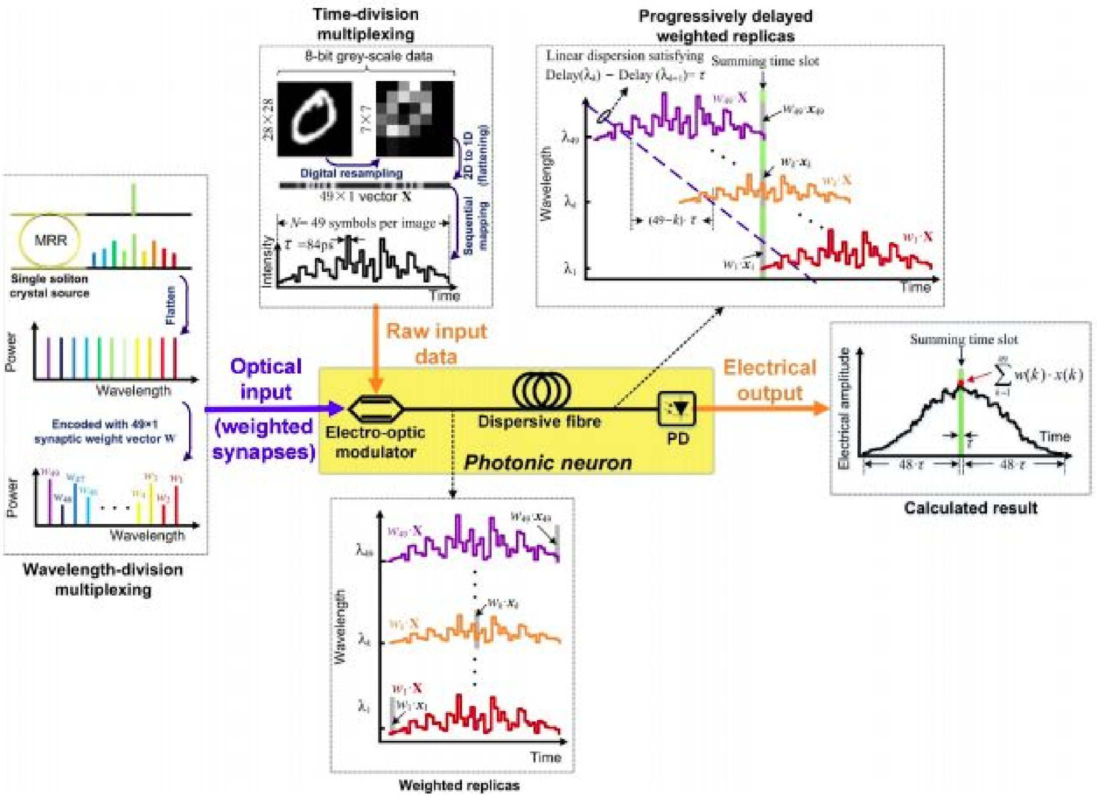


Figure 5: Diagram of the photonic perceptron proposed by Xu et al. Taken from [94]

[70] uses a soliton microcomb to recreate the same temporal data signal on 49 different wavelengths, a waveshaper to scale each of these 49 different wavelengths to the same output intensity, and then a further wavelength-dependent scaling of intensity is performed by a trainable, reprogrammable weight. The resulting 49 different wavelengths, each with the same temporally encoded signal, comprise 49 different intensity values. A second order dispersion effect is then used to progressively delay the different wavelengths, so as to align the diagonal elements of this scaled matrix into a single time-step, which is then detected by an incoherent photodetector. This approach of using the time domain to align scaled input values is unique amongst recent literature, with most research interest focussing on lower latency approaches to computation. Ignoring the delay introduced by the time taken to perform the second order dispersion effect, this method of time-domain signal aligning appears to scale poorly with additional wavelength channels. An in-depth analysis is given in Appendix A.

Whilst this encoding and detection schema may not increase in bandwidth with additional WDM channels available, importantly it increases the potential fan-in of a photonic neuron while introducing a minimum number of additional components, namely a single addition wavelength dependent filter.

2.7.2 WDM photonic neural networks

The optical neural network accelerator architecture proposed in [93] utilizes a WDM CW source fed to each section of this network. Wavelength specific MRRs are used to transfer individual wavelengths onto specific waveguides. The intensity of light on these individual waveguides is scaled by a “weighting block” of multiple electrically-tunable MRRs. Multiple MRRs are used to reduce thermal crosstalk and increase the resolution of the scaling of these weighting blocks. This wavelength is then further scaled by the input, which is encoded onto an electrically-tunable MZI. The outputs of these individual waveguides are transmitted to a final waveguide via passive MRRs, where incoherent photodetection is used to sum these different values.

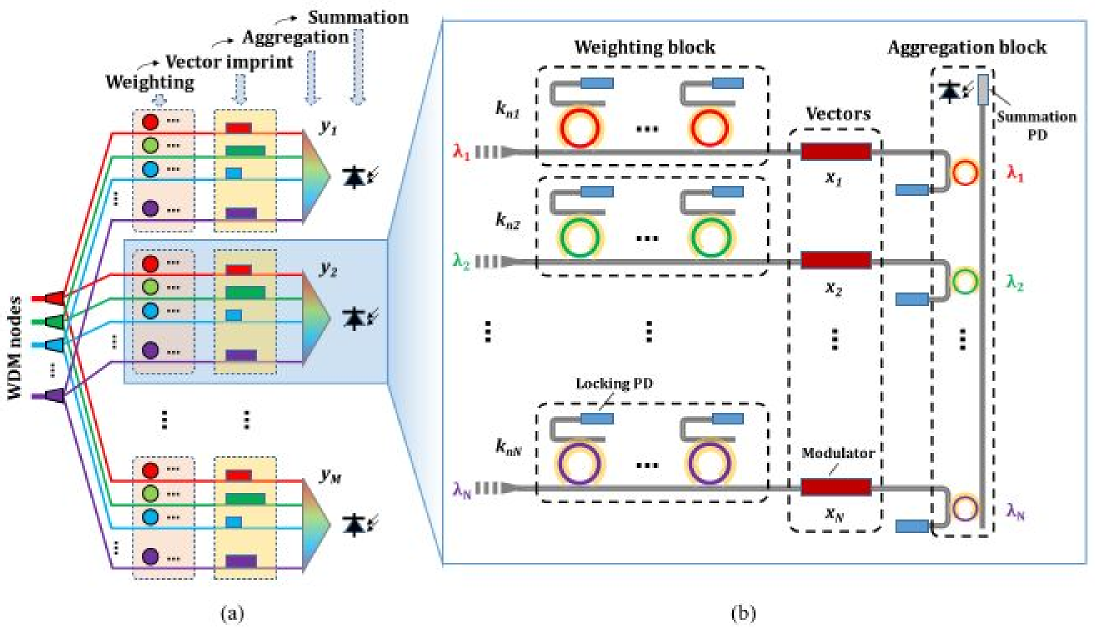


Figure 6: Optical neural network design proposed by Cheng et al. Taken from [93]

As a result, the bandwidth of each ‘neuron’ (photodetector) in this network can be defined as:

$$Bandwidth = \frac{n \cdot r}{t_p + t_{pd} + t_e},$$

where *n* is the number of input wavelengths, *r* is the minimum resolution achievable by the entire system, t_p is the time it takes for optical information to flow from the MRR to the photodetector, t_{pd} is the minimum time slot that the photodetector can operate with, and t_e is the maximum of the time required for the MZM and MRR values to change.

Thus, the bandwidth of the network scales linearly with respect to additional wavelengths, assuming the resolution of the network is held constant.

This architecture requires, per neuron, *k*n* weighting MRRs, *n* tuning MZIs, *n* coupling MRRs and 1 photodetector, to produce 1 weighted sum. This is noticeably more components per calculation than rival proposals, but a considerable focus of the work done in [93] is on improving the available resolution of the weighted sums, by using repeated MRRs on spatially separated waveguides to produce higher resolution outputs.

[93] also investigates the 'double-edged sword' for silicon MRR-based architectures which rely on the thermo-optic effect. Whilst changes to the refractive index, and thus the resonant frequencies, of a MRR by thermo-optic tuning present highly useful intensity scaling and optical switch capabilities, thermal cross-talk between MRRs and a need for fine-grained thermal control of the PIC respective give rise to the issues of uncontrollable, decoupled wavelength control and expensive, complex chip-design additions. A more in-depth analysis can be found in [93].

Similar to the architecture proposed by Cheng et al., [85] proposes an ONN that uses a CW light source comprising multiple different wavelengths into each section of the network. Weighting of the input is performed using a silicon photonic weight bank, and then the actual input for each section is encoded as an electric source connected to a Mach Zehnder modulator. This is then combined using a simple combiner tree with the outputs of other 'synapses' to create a unique, weighted sum of inputs on each of the wavelengths.

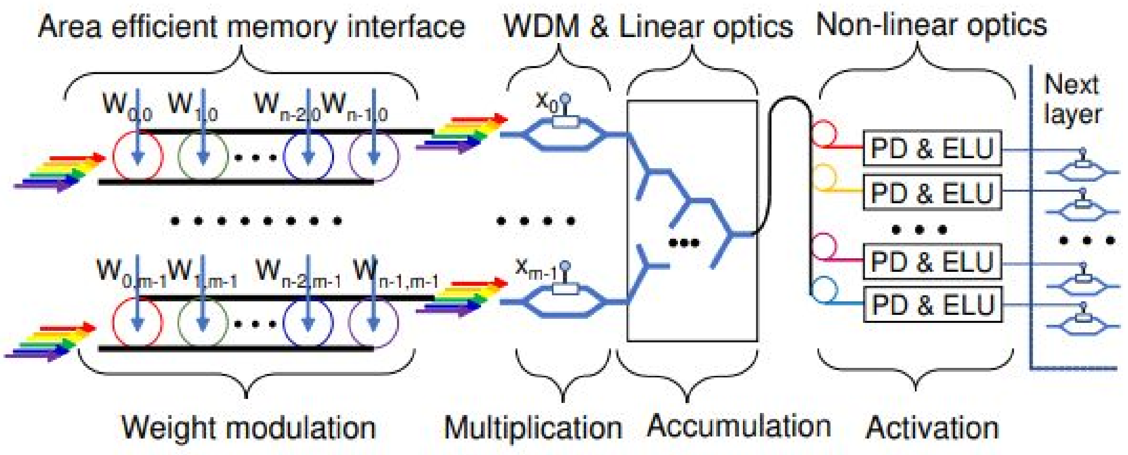


Figure 7: WDM ONN architecture proposed by Ishihara and Shiomi. Taken from [85].

As a result, the bandwidth of this network can be described as:

$$Bandwidth = \frac{n \cdot i \cdot r}{t_p + t_{pd} + t_e},$$

where *i* is the number of inputs (can be equal to or lower than the number of wavelengths),*n* is the number of different wavelengths (and thus MRRs needed at each neuron), *r* is the achievable resolution(ie 3 bit resolution = 2^3 = 8) of a combination of weighting (from 0-1) using a MRR, scaling (from 0-1 for the input) using a MZM and combining using a combiner tree, t_p is the time it takes for optical information to flow from the MRR to the photodetector, t_{pd} is the minimum time slot that the photodetector can operate with, and t_e is the maximum of the time required for the MZM and MRR values to change.

Assuming the values that Ishihara and Shiomi proposed are valid, the denominator of Equation X can cumulatively be less than 100 picoseconds, so 10 layer networks can potentially infer at gigahertz frequencies.

This architecture is very similar to the one proposed in [93], but it has the advantage that the MZMs are reused, as they each scale multiple wavelengths. As a result, this design would require less on-chip space, and less power for heating/exciting the MZMs, although the coherent combiner tree may take up a non-negligible amount of space.

[85] also describe and simulate an opto-electric ELU function, which uses an output from the center of a non-linear sinusoidal electrical circuit as the input to the next MZM, and the simulate the error arising from this approximation as only 2%.
Details about the error arising from using the same MZM for different wavelengths is not provided.

Using MZMs for multiple wavelengths introduces some inherent reduction in resolution of the input encoding: using the Pockels effect, the phase difference in a MZM can be calculated as

$$2\phi = 2\left(\frac{2\pi}{\lambda}\right)\left(\frac{cVL}{d}\right). \quad \textbf{(3),}$$

where *V* is the voltage applied, *c* is the speed of light, *L* is the length of the MZM and *d* is the width of the MZM arms.
With a constant voltage, and all other parameters being held constant (as they are component dependent parameters, and do not change with respect to the environment), the phase shift will be inversely, linearly proportional to the wavelength according to Equation 3. However, with a properly designed MZM, this effect can be minimised. In addition, a larger MZM here may take up more space, but since it is used to perform n distinct scalings of n different wavelengths simultaneously, the increased size of the MZM is somewhat abated.

[95] proposes use of a silicon photonic weight bank [96][97] and the excitable dynamics of a sub-threshold pumped laser attached to a WDM waveguide that utilises the Broadcast-and-Weight protocol to create highly interconnected networks of spiking neurons. Use of WDM signals in a single waveguide reduces the need for excess neuron interconnects via spatially multiplexed waveguides.

A potential disadvantage to the Broadcast-and-Weight protocol is that either MRR weighting of signals at each silicon weight bank alongside the shared waveguide must be artificially limited to prevent a single neuron potentially absorbing all of a single wavelength, or incoming signals are split from the shared waveguide, reducing the intensity and thus resolution of the MRR scaling. The other option is that each weight bank has the ability to strip multiple wavelengths from the waveguide, resulting in neurons further along the propagation path of the waveguide virtually being connected to fewer neurons. This may not necessarily be a disadvantage, provided that this Winner-Take-All analogous functioning is properly accounted for in network design and training, but it does limit either the reconfigurability or resolution of the network.

Each ‘neuron’ is made up of an excitable laser, two silicon weight banks comprising tunable MRRs, two photodetectors and a passive decoupling double MRR. One weight bank is labelled ‘excitatory’, the other ‘inhibitory’, as their WDM scaled outputs are incoherently detected by their respective two photodetectors, with the inhibitory photodetector placed behind the excitatory photodetector. The laser is connected to their midpoint, such that the subtraction of the current produced by the inhibitory light intensity from the current produced by the excitatory light intensity occurs, and potentially excites the laser into producing a spike.
5.1 effective bits of control resolution is experimentally demonstrated in [98], but as discussed in [93],altering the resonance spectrum of cascading individual MRRs attached to a common waveguide will reduce their available tuning range, as MRRs attached to the same bus must exist within one free-spectral range (FSR), to avoid inter-channel crosstalk via unintended MRR tuning. A potential method to alleviate this is to first passively couple each signal onto a spatially-separate waveguide before tuning a single wavelength-specific MRR, and then optically combining these photons, as proposed in [93].The obvious disadvantage of this approach is an increase in the number of components and thus a decrease in neuron density, although given the relative size of waveguides and MRRs with respect to excitable lasers, this may be an agreeable tradeoff.

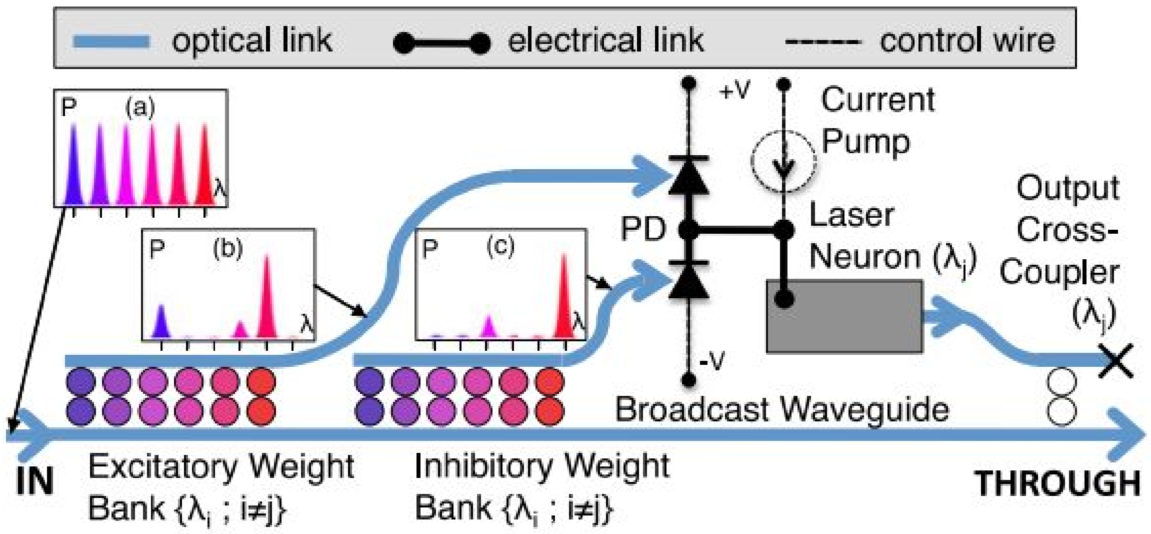

optical link
electrical link
control wire
P
(a)
λ
P
(b)
λ
P
(c)
λ
+V
-V
Current Pump
PD
Laser Neuron (λ_j)
Output Cross-Coupler (λ_j)
Broadcast Waveguide
IN
Excitatory Weight Bank {λ_i ; i≠j}
Inhibitory Weight Bank {λ_i ; i≠j}
THROUGH

Figure 8: Photonic spiking neuron employing the Broadcast-and-Weight protocol, proposed by Tait et al. Taken from [95].

‘Photonic Modulator Neurons’ proposed in [92] employ off-chip, CW light sources and seek to modulate these sources electro-optically using light that has been weighted by an internal silicon photonic weight bank[96][97]. Output light from layer *i-1* encodes information in the intensity of WDM signals which are split using optical splitters into *n* separate waveguides of equal intensity. Light on each separate waveguide is coupled onto another waveguide via a photonic weight bank before being fed into an excitatory photodetector, while the uncoupled light on the original waveguide is fed to an inhibitory photodetector as in [95].
The output electrical current electro-optically tunes a MRR that scales the CW light in the next section.

An advantage of this idea is that the light intensity at each neuron or neuron layer can be considered to be constant, and equivalent to the off-chip light source's input power, minus insertion losses. The downside comes in the loss of spike-processing and thus temporal coding abilities, as well as the increased energy requirements in pumping each section of the PIC with CW light.

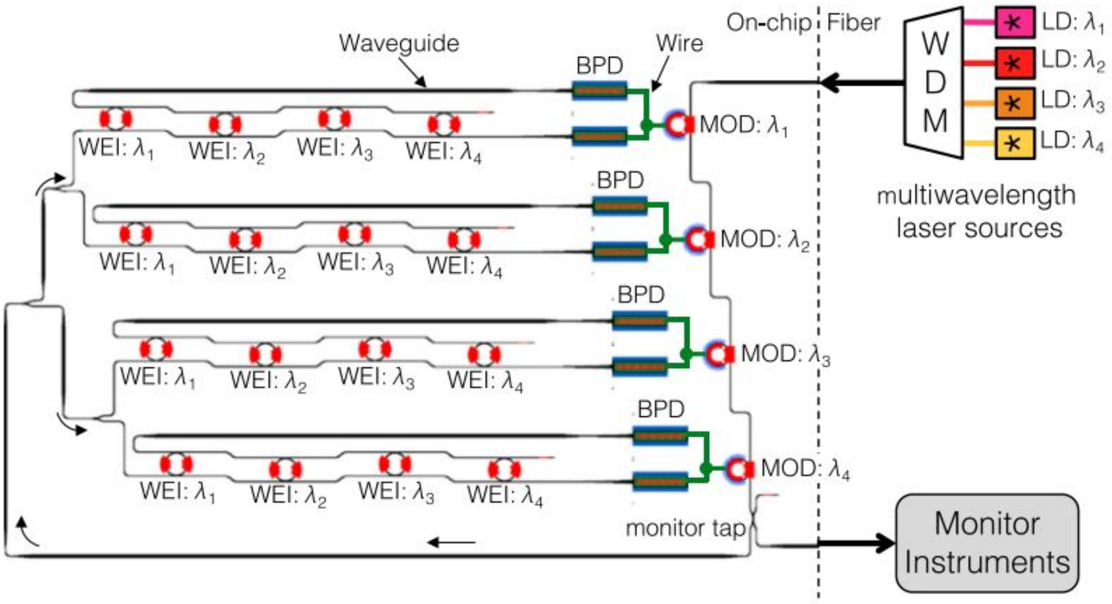


Figure 9: An example photonic modulator network which employs the Broadcast-and-Weight protocol comprising 4 neurons, with monitor instruments for experimental validation. Taken from [92]

The bandwidth of a layer of this network can be considered to be:

$$Bandwidth = \frac{n \cdot i \cdot r}{t_p + t_{pd} + t_e}$$

, where *n* is the number of independent input wavelengths, *i* is the number of output MRRs (which tune the next layer), *r* is the achievable resolution(ie for 4 bits of resolution, $r = 2^4 = 16$), t_p is the time taken for light to propagate from a layer to the photodiode, t_{pd} is the minimum photodetector sampling time, and t_e is the time taken for the electric circuit effects, including electro-optic tuning, to take place.

Provided that a MRR can be tuned electro-optically with the same resolution as an MZI, this architecture results in the same bandwidth as the architecture proposed in [85]. Importantly however, it removes the need for the relatively bulky MZIs.

[99] builds a theoretical coupled-mode theory of MRRs to use negative spikes (perturbations from steady-input signal) to perform spiking network calculations on MRRs. Cascadability of spikes between two excited MRRs is simulated, with the existence of a cascaded spike taken as evidence of cascadability of signals. Whilst this is technically true, and the excitation of a second spike by a first spike is simulated, the second spike exists for a

shorter time period than the first, indicating that time-domain scaling and thus a loss of signal integrity is displayed through subsequent ‘neurons’.
Additionally, this proposal claims to be passive, yet requires a steady pump CW wave to each ‘passive’ component to remain at the excitable state. If the passive MRRs and nanobeams are not kept optically pumped, and instead are optically pumped by input signals, then loss of signal across each component will likely render the signal too small to excite a latter neuron after only a few propagations. Additionally, the time-domain shrinking of the second spike simulated indicates that infinite cascadability is not realisable using these negative-spiking MRR and nanobeam neurons.

2.7.3 Coherent photonic neural networks

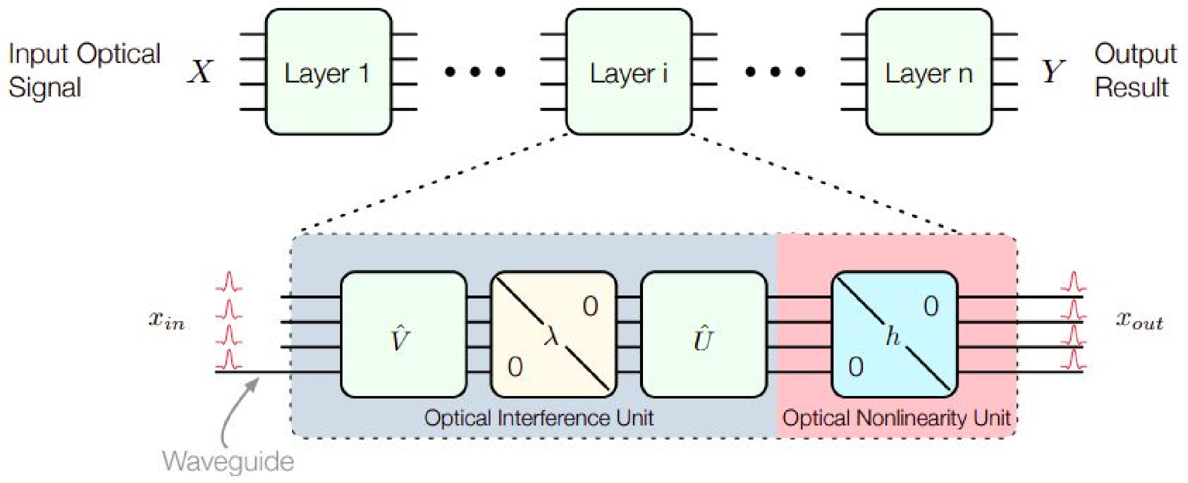
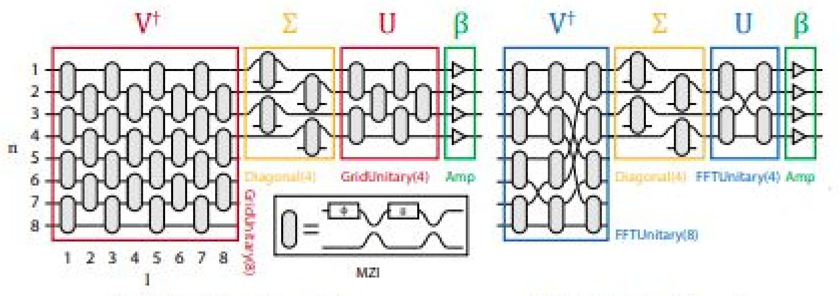


Figure 10: Diagram of the proposed optical neural network outlined in [83]

In [83], an optical neural network is realised via a densely interconnected mesh of MZIs, which comprise an Optical Interference Unit (OIU) which performs a matrix multiplication and an Optical Nonlinearity Unit (ONU), which is simulated electronically. The matrix multiplication is performed using a coherent light source of variable intensity, which is fed into an interconnecting mesh of MZIs. Singular value decomposition theory posits that any real valued matrix multiplication can be reduced into a rotation (which can be performed by MZIs), then a diagonal scaling (performed by optical amplifiers), and then another rotation, to convert back to the original basis. The first and last rotations can be done with MZIs. As a result, this paper demonstrated that arbitrary matrix multiplication can be performed with a sufficiently large number of MZIs and optical amplifiers. More specifically, for 4 layers of 4 neurons 56 MZIs are required. The latency of the network is just the time of flight of the photon, provided that the theorised ONU can be implemented on-chip via saturable absorption or another purely optical method. If the ONU is not realisable by purely optical means, then the latency of this system drops substantially, as each layer will induce an additional latency at the non-linear activation function.



((a)) GridNet linear layer ((b)) FFTNet linear layer

Figure 11: The proposed linear layer architectures for the GridNet and FFTNet optical networks simulated in [91]

[91] extends the work done in [83] with an analysis of different layouts for the MZI mesh. They also measured success based upon the ONN's success at a predetermined task, not just based upon the fidelity of signals transformed by the MZI mesh against the simulated output. Their work explores 2 main MZI mesh variants, namely GridNet and FFTNet, in detail, and assesses their performance and sensitivity to noise. GridNet is a fully connected mesh of MZIs, and exhibits a higher initial accuracy if no component noise is simulated, but drops quickly below the results of FFTNet. FTTNet, which lacked the full computational capability of GridNet, maintained the same accuracy on the given task with a quantization to 6 bits per weight. FFTNet also maintained performance over a reasonable amount of component imprecision and noise, up to 2% noise in component signals.

[84] also looks to improve the work done in [83], by introducing a reconfigurable nonlinear activation function which functions via electro-optic modulation of intensity on a MZI. 93% test accuracy on the MNIST [55], [83] dataset is simulated, using a fully connected network operating on Fourier space information, and a simulated linear network achieves 85% test accuracy. By operating in the Fourier domain, their proposed network is parameter efficient, requiring only $2 \times N^2 \times L$ learned parameters, where N is the number of complex Fourier domain inputs and L is the number of layers in the network. In comparison to a standard 2 layer, fully-connected network operating on the MNIST dataset, which would have 28 x 28 = 784 artificial neurons in the first layer, their proposed design uses only 16 input values and achieves a commendable 93% test accuracy.

In [100], a theoretical implementation of using optical signals to encode both information and weights is proposed to perform matrix multiplications before being reconverted into the photonic domain. Weighting of input data occurs via interference with the coherent weight signals, before being integrated and an electrical nonlinear function is applied. It is proposed to do this using free-space optics, and the authors believe that sub-pJ energy expended per multiply-accumulate operation performance is achievable.

2.7.4 Training

The training of deep neural networks currently presents a bottleneck to research and software deployment, and also requires substantial energy expenditure.

An important requirement of most neural-network dedicated hardware accelerator chips is that on-chip training, usually performed via updating weights by gradient descent performed upon a back propagated signal of errors, is easily available. On-chip implementation of backpropagation would likely enhance industry adoption of PICs for neuromorphic computations, and additionally it may help abate the issues faced by the relatively large size of photonic components. Small SWaP (Size,Weight and Power) constraints may limit the viability of PICs for edge computing, due to their size, but server-side and industrial use cases would likely increase if on-chip training can provide lower energy expenditures and higher throughputs, albeit at larger chip-sizes.

One potential application for a high-throughput, area-hungry neural network would be in automated content moderation for social media networks and online video uploading websites. These websites have millions of images and hundreds of thousands of hours of video uploaded to them each day, and so it is important that automated content moderation be performed on all uploaded imagery as fast and energy-efficiently as possible.

A major gap in the majority of aforementioned optical neural network architectures is the apparent lack of ability of on-chip training. [101] utilises the adjoint variable method to perform on-chip backpropagation of errors via physically propagating photons. Use of the finite difference method and forward propagation is proposed by [83] as a method of discovering on-chip the contributions of different components, and thus their derivatives, to the error of the network. This method scales linearly with the number of parameters in the network, which in neural networks can number in the billions, and as such is untenable for most real-world use, regardless of the speed of the substrate on which the computations occur. Simulations performed in [102] on a traditional computer indicate that gradient descent of weights via backpropagation of errors adjusted with respect to photonic noise, weight limits, activation differences and initialisation schema can yield better results than ignoring these fundamental system constraints, but did not go so far as to train an optical neural network on-chip.

To the author’s knowledge, there exist no proposed methods yet that can perform backpropagation of errors on a WDM-based architecture on-chip.

2.8 Comparison between selected photonic neural network architectures

Since differences exist in how the proposed and implemented architectures compute information, and a direct neuron isomorphism is not always given or easily created, comparisons between architectures and designs are not always linear. The estimation of area of the architecture and latency, as well as the energy consumption per bit will be reduced to qualitative comparisons and descriptions, and not quantitative comparisons, as there is insufficient information available. Relative and absolute measurements of the throughputs of each architecture are not included, as can be dependent upon the auxiliary electronics, the number of sequential layers without digital sampling and numerous other design considerations, which are not fully formed and defined in the proposed and implemented optical network architectures.

Table 3: Comparison table between implemented and proposed fully-connected photonic ANN architectures

Research literature	Physically implemented	Latency per MAC	Energy consumed per calculation	Resolution of calculations	Relative estimated area of architecture	Type of nonlinearity	Optoelectronic conversion scheme(where applicable)
[85]	No. Simulated results available.	<100 ps	Not reported. Ignoring electro-optic conversion power(amortized over multiple spatially multiplexed components receiving the same CW WDM input signals), and stationary thermal-tuning of MZIs and MRRs, energy required can be theorised to be on the order of tens of fJ per calculation.		Medium, due to incorporation of *i* MZIs for *i* different inputs, but uses MRRs for individual wavelength weighting.	Electrical circuit based (architecture proposed utilised the ELU activation function), powered by photodetector	Balanced photodetector (compares signal against reference signal) for each individual wavelength
	Yes, using desktop available components. No on-chip implementation as yet	Demonstrated single perceptron latency: 64 μs Achievable (with concentrated second order-dispersion): 8.64 μs	Not reported	Reported 8 bits of information per wavelength (49 wavelengths demonstrated)	Proposed on chip implementation: Medium, due to size of integrated LiNb modulators and second-order dispersive elements	Non-linearity not implemented/ discussed	Time-sampled photodetection (limited temporal window comprising valid sample of calculation is available, rest is unusable)
[92]	Yes	Assumed <100ps per MAC (CW signal propagates across tuned MRRs, into balanced photodetectors before electrical non-linearity	Not reported. Ignoring electro-optic conversion power(amortized over multiple spatially multiplexed components receiving the same CW WDM input signals), and stationary thermal-tuning of	Not reported, assumed to be limited by MRR silicon weight bank resolution (5 bits)	Small - comprises only spatially separate waveguides, MRRs and photodetectors	Electro-optic (electric-circuit based)	Balanced photodetection of incoherent WDM signals used

		is applied to then tune a modulating MRR)	MRRs, energy required can be theorised to be on the order of ones of fJ per calculation.				
	No	Assumed <100ps per MAC (CW signal propagates across tuned MRRs, an MZI and then is detected by a photodetecto r)	Not reported. Ignoring energy required for electro-optic conversion(amorti zed over multiple spatially multiplexed components receiving the same CW WDM input signals), the energy consumed is by tuning of an MZI and k MRRs. On the order of tens to hundreds of fJ per calculation.	Not reported, but can be assumed to be higher than 5 bits[] due to reduced variability from thermal component crosstalk	Medium, due to the need for an MZI, multiple tunable MRRs and 2 coupling MRRs per wavelength per MAC	Electro-optic (electric circuit based)	Incoherent photodetection of multiple WDM signals
[83], [91]	Partially - optical matrix multiplication is demonstrated, optical nonlinearity is electronically simulated	Tens to hundreds of picoseconds, depending on the number of layers in the network. Latency is limited only by the time of flight of the photon through the MZI and waveguides, and across the proposed optical nonlinearity.	Shen et al. estimate the power consumed to be ~5/*mN* fJ per operation, where m is the number of layers of N*N matrix multiplications. This doesn't take into account the cost of tuning each MZI, which is ~10mW on average.	Not reported. Sequential MZI errors will be the limiting factor, as phase noise compounds sequentially.	Large: WDM signals not used to increase spatial efficiency, and MZIs used	Optical nonlinearity is proposed, simulated on a computer	Coherent photodetection
[100]	No	Signal pulses larger than 100ps are assumed, and propagation and electrical delays would increase this to	Authors theorise that a few fJ per MAC is possible, provided large batch-sizes are utilised(to amortize the cost of electro-optic conversions to produce input and weight signals)	Not reported	Large: WDM signals not used, and free space optics assumed, resulting in large-area consumption	Electrical circuit based	Balanced homodyne detection used

Table 4: Analysis and comparison of proposed and realised photonic convolutional neural network architectures

Paper	Physically Implemented	Latency of a single convolved pixel	Resolution	Relative architecture area	Nonlinearity	Electro-optic conversion scheme
DEAP-CNNs [103]	No. Simulated performance evaluated.	200ps (limited by ADCs/DACs)	7 bits theorised	Small: MRR weights modulate CW input and MRRs weight signals, balanced photodetection	Analog electrical nonlinearity via TIA	Balanced photodetection
A Winograd based Integrated Photonics Accelerator for Convolutional Neural Networks [104]	No. In-house tool for simulation built, not open-sourced	200ps (limited by electrical clock signal)	6 bits (due to memristor accuracy)	Small: MRRs proposed for weighting inputs	Analog electrical nonlinearity	Incoherent unbalanced photodetection
On-Chip Optical Convolutional Neural Networks [105]	No. Simulated, with errors based off of the work done by Shen et al.	Not mentioned. Assumed to be 200ps (limited by ADC clock speed).	8-bits theorised	Large: MZI mesh proposed for weighting inputs	Electrical nonlinearities (analog ReLU) or optical nonlinearities (graphene saturable absorption) proposed	Photodetection
PCNNA [106]	No. Theoretical analysis given, no simulation of results.	200 ps (5 GHz clock runs the optical components)	Assumed 7 bits, same as [103]	Small: MRRs for weighting	Not discussed	Not explicitly mentioned. Assumed balanced photodetection (based on Fig 1. in [106])
HolyLight-M [20]	No. Simulation of photonic components performed.	1.28 Ghz (limited by ADC) sampling, which corresponds to 781ps.	16 bits	Large: 280.42 mm^2 according to Liu et al. Matrix vector multiplication performed by splitting input into multiple 2 bit calculations	Electronic digital nonlinearity	Unbalanced photodetection
HolyLight-A [20]	No. Simulation of photonic components performed.	12.8Ghz (78 ps)	16 bits	Medium: 22.46 mm^2 according to Liu et al. Binary	Electronic digital nonlinearity	Unbalanced photodetection

				addition performed by bit-shifting, tightly packed MRR crossbar arrays. Removed need for ADC by using binary MRR control, which simplifies interfacing electronics and increases operating speed.		
Parallel convolution processing using an integrated photonic tensor core [107]	Yes. 4 kernels and 4 unique WDM signals are convolved (each signal is convolved by one kernel)	74 ps (clear eye opening at 13.5 GHz)	5 bits	Small: microring resonator crossbar array similar to (cite silicon microring crossbar array), and microresonator used for frequency comb generation	Not discussed.	Unbalanced photodetection. 200 kHz photodetector (New Focus Model 2011) for experimental convolution results, 12 Ghz photodetector used for frequency response tests.

2.9 Literature Review Conclusion

It is almost self-evident that the current trend in the field of deep learning, of throwing ever larger amounts of computational resources at a problem, is unfeasible. Traditional digital electronic CPUs are ill-suited to the specific workloads of modern scientific and deep learning calculations, and dedicated hardware co-processors designed for deep learning which use digital electronics may not solve the issue.

Renewed funding and research in silicon photonics provides a desirable and potentially useful avenue of progress, to tackle the excess energy consumption of digital electronics whilst also likely increasing the throughput and decreasing the latency of calculations. The path forward is not all smooth-sailing though; experimental realisations of photonic neuromorphic hardware pale in comparison to the sizes of neural networks trained on traditional computers, and the relative inflexibility of analogue photonic hardware in comparison to digital electronics is a considerable point of friction.

Monolithic integration of photonics and electronics poses arguably the best path forward, with high-speed dedicated hardware to modulate electrical signals into the optical domain and pass information on to more passive photonics processing elements, but will require years and mountainous amounts of industry research to come to fruition.

Photonic neuromorphics is thus an exciting avenue of further research, and the next large steps to be taken must involve scaling up the magnitude of implementations to be able to provide more evidence of the advantages of silicon photonics over digital electronics on real world use cases.

3. Proposal for a Feedforward WDM Optical Neural Networks with Analogue Electronic Activation and physically backpropagating error signals for on-chip training

3.1 Introduction

This section of the document intends to present to the reader a complete, simple mathematical explanation and derivation of the formulae required to update the weights in an artificial neural network via stochastic gradient descent. Once this is complete, a detailed proposal for an on-chip photonic neuron that uses high-speed analog electrical circuits and physically analog photonic neurons is presented, which outlines the requirements for building a feedforward photonic neuron and the additional requirements for on-chip learning by stochastic gradient descent on physically back-propagated error derivatives. Then, a consideration of the size, components and is performed. Lastly, the roadblocks as perceived by the author are introduced, and many of these are left to further work in both industry and academia to solve.

3.2 A Mathematical Exploration of Stochastic Gradient Descent by back propagation of errors in Deep ReLU Networks

The traditional, continuous valued artificial neural network (ANN) is based upon a sequential network of parallel “neurons”. Each “neuron”, which is loosely based upon the neurons in a biological brain, can transmit a signal to other neurons, depending upon its current state and the inputs it receives. In its simplest form, a neuron will multiply each numerical input by some linear weight, add a bias and then apply a non-linear activation function. By utilising a non-linear activation function, neural networks can learn to map non-linearly separable inputs to desired outputs.

The output of each neuron ***y*** is described in Equation 1, where ***f()*** is a non-linear activation function, ***x*** is a vector of *k* inputs, ***b*** is a bias and ***w*** is a vector of *n* weights.

$$y = f\left(\left(\sum_{i=0}^{k} w_i \cdot x_i\right) + b\right) \qquad \textbf{(1)}$$

Feedforward artificial neural networks are often abstracted into layers, where each layer of neurons receives input from the prior layer, and propagates its outputs to the next layer of neurons.

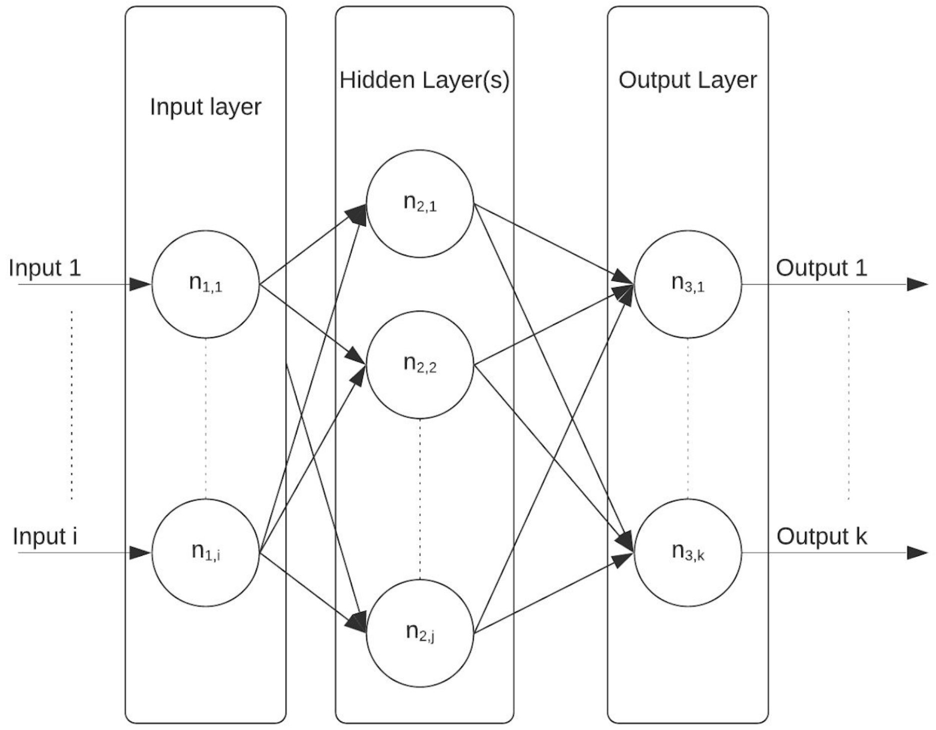


Figure 12: Abstract diagram of a multi-layer feedforward neural network

We shall define some mathematical notations before proceeding. We shall consider some intermediate point in a network, either a neuron or an output of a neuron, as $x_{a,b}$, where x is the intermediate point in layer a in position b. To further clarify this, we define the weighted sum of inputs, plus a bias, of neuron b in layer a as $n_{a,b}$. We also define the non-linear activated output of a neuron b in layer a as $z_{a,b}$. Similarly, we define the weight $W_{a,b,c}$ as the weight applied to the input c by the neuron b in layer a.

Each neuron $n_{n,m}$ computes the non-linear activation of the weighted sum of its inputs plus some bias, according to Equation 1. This value is then propagated forwards to the next layer, where again each neuron computes the non-linear activation of the weighted sum of its inputs plus some bias.

In order to adjust these weights and biases in a manner more time efficient than simulated annealing or genetic algorithms, we will perform Stochastic Gradient Descent (SGD) via backpropated error signals. The weight update rule for SGD is as Equation 2:

$$\Delta W_{n,m,p} = -n \cdot \frac{\partial L}{\partial W_{n,m,p}} \tag{2}$$

And the bias update rule, similarly, is shown in Equation 3.

$$\Delta B_{n,m,p} = -n \cdot \frac{\partial L}{\partial B_{n,m,p}} \tag{3}$$

In the above SGD-update rules, the learning rate *n* is an externally set hyperparameter, equivalent to the size of the 'steps' down the 'hill' that we are taking. The *L* value is the loss function of the network, which is the function that we are seeking to minimise. By updating the weights and biases according to the SGD update rules above, we are moving in a direction which minimises this loss function, albeit in tiny steps.

These partial derivatives can be calculated using the chain-rule iteratively for each layer. For example, the partial derivative of the loss with respect to $W_{3,1,2}$ (i.e. the weight of the second input to the neuron $n_{3,1}$) can be written as:

$$\frac{\partial L}{\partial W_{3,1,2}} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial z_{3,1}} \cdot \frac{\partial z_{3,1}}{\partial W_{3,1,2}} \tag{4}$$

where: $\frac{\partial L}{\partial y}$ is the partial derivative of the loss function with respect to the combined network output *y*, $\frac{\partial z_{3,1}}{\partial W_{3,1,2}}$ is the derivative of the output ($z_{3,1}$) of neuron $n_{3,1}$ with respect to the weight $W_{3,1,2}$, and $\frac{\partial y}{\partial z_{3,1}}$ is the derivative of the network output *y* with respect to the output of neuron $n_{3,1}$.

The derivative $\frac{\partial L}{\partial y}$ is a constant for all neurons in the network, and $\frac{\partial z_{n,m}}{\partial W_{n,m,p}}$ is the partial derivative of some neuron *m* in layer *n*'s output with respect to the weight *p* is simply $x_{n,m,p}$ multiplied by the derivative of the output of the non-linear activation function with respect to the neuron's weighted sum, as shown in Equation 5, with ***f*** being the nonlinear activation function of that neuron.

$$\frac{\partial z_{n,m}}{\partial W_{n,m,p}} = \frac{\partial f}{\partial \sum_{i=0}^{j} \left(W_{n,m,i} \cdot x_{n,m,i} \right) + b} \cdot \frac{\partial \sum_{i=0}^{j} \left(W_{n,m,i} \cdot x_{n,m,i} \right) + b}{\partial W_{n,m,p}}$$

$$\frac{\partial z_{n,m}}{\partial W_{n,m,p}} = \frac{\partial f}{\partial \sum_{i=0}^{j} \left(W_{n,m,i} \cdot x_{n,m,i} \right) + b} \cdot x_{n,m,p}$$

(5)

The extended form for the derivative of the loss function with respect to the bias of any neuron $n_{n,m,p}$ is similarly described in Equation 6.

$$\frac{\partial z_{n,m}}{\partial B_{n,m}} = \frac{\partial f}{\partial \sum_{i=0}^{j} \left(w_{n,m,i} \cdot x_{n,m,i} \right) + b_{n,m}} \cdot \frac{\partial \sum_{i=0}^{j} \left(w_{n,m,i} \cdot x_{n,m,i} \right) + b_{n,m}}{\partial b_{n,m}}$$

$$\frac{\partial z_{n,m}}{\partial B_{n,m}} = \frac{\partial f}{\partial \sum_{i=0}^{j} \left(w_{n,m,i} \cdot x_{n,m,i} \right) + b_{n,m}}$$

(6)

The ReLU (Rectified Linear Unit) activation function was chosen for this proposed photonic-electric neuron, due to the three following reasons:

- It is simple to understand and has a simple derivative, which will reduce the complexity of the analog electric circuits still to be designed. The derivative of the ReLU function is the Heaviside Step Function.
- It is inspired by neuroscience, and is arguably the most biologically plausible threshold function used in discrete-time (non-spiking) artificial neural networks.
- Owing partially to its computational simplicity and partially to its commendable performance, it is widely used across a number of deep-learning tasks, and is generally considered the most common activation function in use today.

- Better convergence than sigmoid and tanh nonlinearities, as a result of the linear scaling of back propagated signals, which prevents gradient extinction.

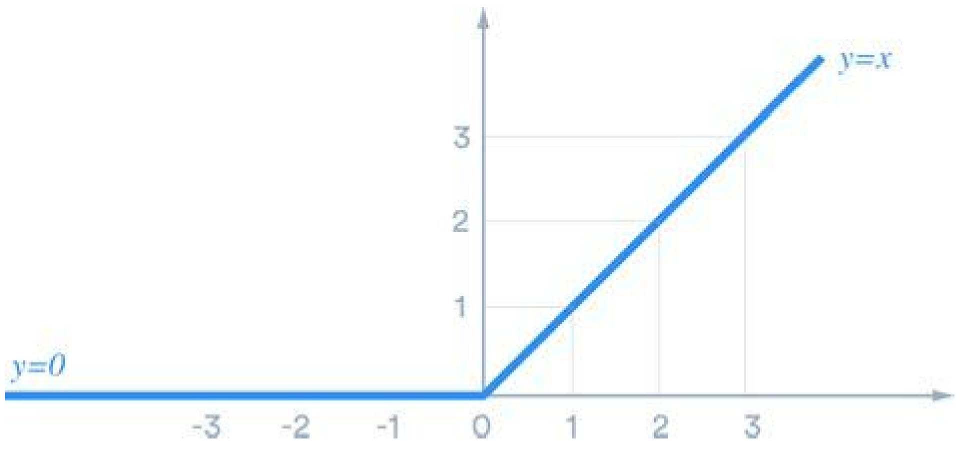


Figure 13: Rectified Linear Unit cartesian representation. At the point x=0, the derivative is taken to be 0, such that the derivative for all x is 0 when the x is less than or equal to zero, and 1 otherwise.

Using the ReLU activation function, the weight update and bias update circuits are greatly simplified, as the derivative of the ReLU function is either zero or 1 for all values of x.

Rewriting the weight update and bias update rules to specifically use the ReLU function as the activation function, we arrive at Equations 7 and 8 below.

$$\frac{\partial z_{n,m}}{\partial B_{n,m}} = \begin{cases} 1 \ if \ z_{n,m} > 0 \\ 0 \ otherwise \end{cases} \quad (7)$$

$$\frac{\partial z_{n,m}}{\partial W_{n,m,p}} = \begin{cases} x_{n,m,p} & if \ z_{n,m} > 0 \\ 0 & otherwise \end{cases} \quad (8)$$

With the above definitions, the only additional calculation needed is the derivative of the output of the network with respect to the neuron to be updated, which is the value of $\frac{\partial y}{\partial z_{n,m}}$.

Since the vector *y* is defined (in the network shown above) as the sum of the outputs of the neurons in the final layer, the derivative of the output with respect to neurons in this layer is simply:

$$\frac{\partial y}{\partial z_{n,m}} = \begin{cases} 1 \ if \ z_{n,m} > 0 \\ 0 \ otherwise \end{cases} \quad (9)$$

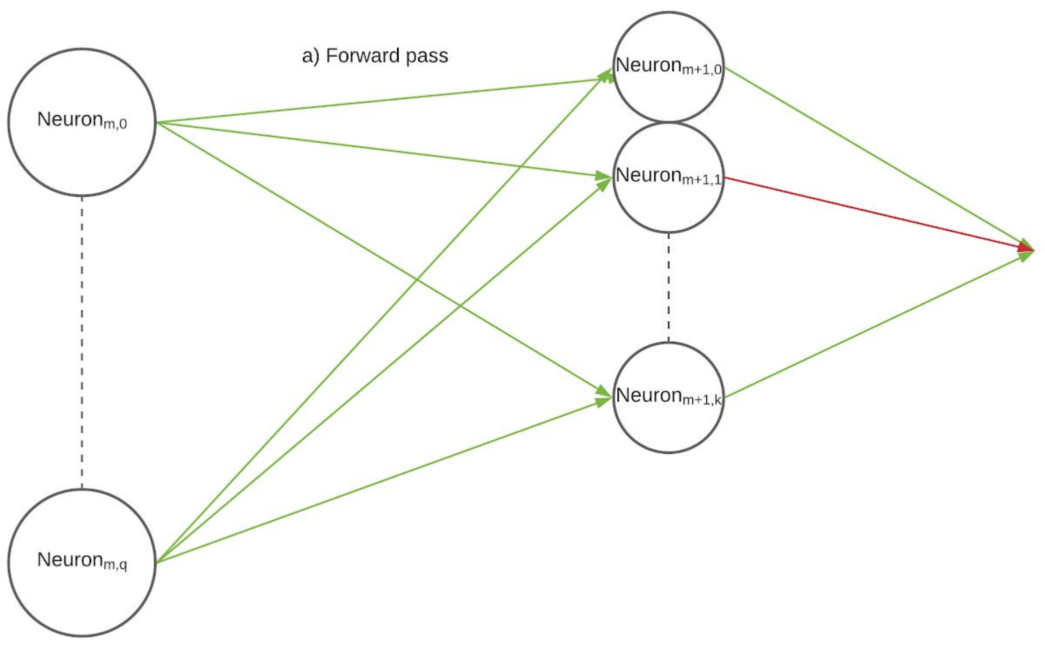
For neurons in the prior layers, it gets a little more complicated. It may be conceptually easier to imagine a single neuron in the second last layer. The derivative of the output *y* with respect to an earlier neuron's output can be shown as:

$$\frac{\partial y}{\partial z_{m,q}} = \sum_{k=0}^{j} \frac{\partial y}{\partial z_{m+1,k}} \cdot \frac{\partial z_{m+1,k}}{\partial z_{m,q}}$$

$$\frac{\partial y}{\partial z_{m,q}} = \sum_{k=0}^{j} \left(\frac{\partial y}{\partial z_{m+1,k}} \cdot \begin{cases} w_{m+1,k,q} & if\ z_{m+1,k} > 0 \\ 0\ otherwise \end{cases} \right)$$

(10)

To put this somewhat lengthy equation into more succinct terms, the derivative with respect to the total network output *y* of a neuron *q* in layer *m*'s output is equivalent to to the sum of the derivatives with respect to the output of all of the neuron in layer *m+1*'s outputs multiplied by their weighting of the neuron *q*'s output.



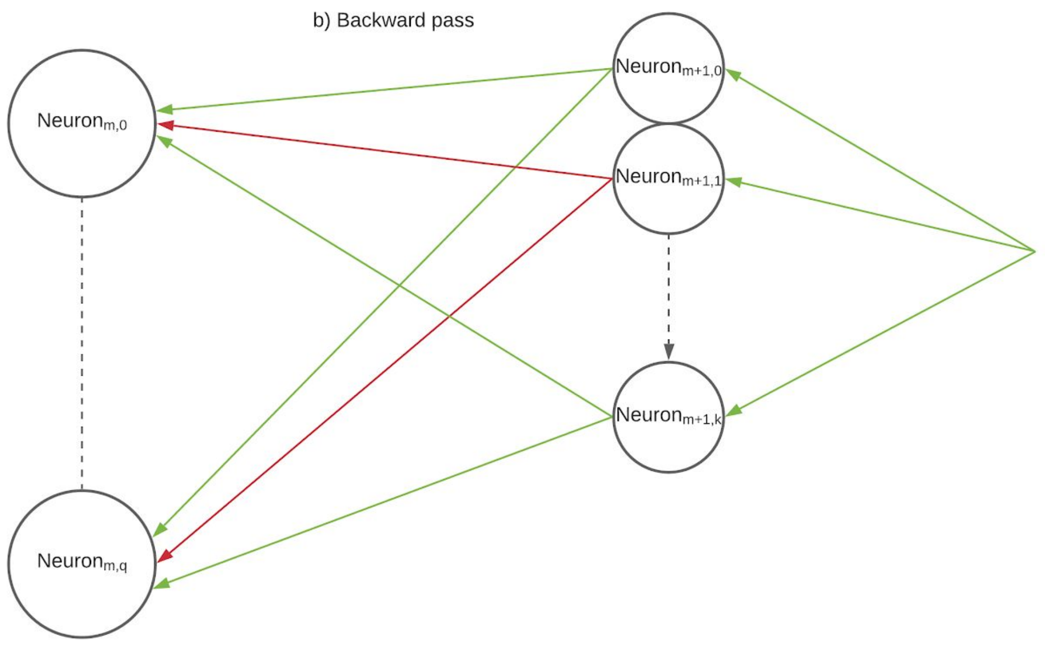


Figure 14: Diagram of signals, represented as arrows, propagating through a neural network. Green arrows represent a value greater than zero, while red arrows indicate a value of zero. a) Signals pass from layer *m* to layer *m+1* in the forward pass, and neuron$_{m+1,1}$ is not stimulated enough to output a value greater than zero. b) In the backwards propagation pass, signals propagate from layer *m+1* to layer *m*. Neuron$_{m+1,1}$ did not propagate a signal greater than zero forwards, and so does not propagate a signal back to any of the neurons in layer *m*.

Another way of explaining this equation is as follows: a neuron *k* in the layer *m+1* will propagate a derivative back to all the neurons in layer *m* that it is connected to, if and only if the neuron *k* had an output greater than zero. The signal propagated backwards from neuron *k* to a neuron *q* in layer *m* will be scaled by the weight that neuron *k* applies to the input it receives in the forward propagating pass from neuron *q*. Thus, from neuron *q*'s point of view, the back propagated signal it receives is equal to the sum of all signals received from all the neurons in layer *m+1*, and each individual signal is scaled by the respective weight applied to the output of neuron *q.*

As such, if we imagine a physical error signal propagating backwards through a dedicated, physical analog network, each neuron in a layer needs to only scale the local backpropagating error signal by the weights it applies to each incoming, forward propagating signal.

Then, each neuron need not have to compute the derivative of the loss function with respect to its output, as it has access to the total local error derivative $\frac{\partial L}{\partial z_{n,m}}$ and so can apply the local weight and bias update rules, as shown in Equation 11. Substituting the local weight derivative and bias derivatives, namely the results of equation 7 and 8, we arrive at Equation 12 and 13.

$$\frac{\partial L}{\partial W_{n,m,p}} = \frac{\partial L}{\partial z_{n,m}} \cdot \frac{\partial z_{n,m}}{\partial w_{n,m,p}} \tag{11}$$

$$\frac{\partial L}{\partial W_{n,m,p}} = \frac{\partial L}{\partial z_{n,m}} \cdot \frac{\partial z_{n,m}}{\partial W_{n,m,p}} = \begin{cases} \frac{\partial L}{\partial z_{n,m}} \cdot x_{n,m,p} \; if \; z_{n,m} > 0 \\ 0 \; otherwise \end{cases} \tag{12}$$

$$\frac{\partial L}{\partial B_{n,m}} = \frac{\partial L}{\partial z_{n,m}} \cdot \frac{\partial z_{n,m}}{\partial B_{n,m}} = \begin{cases} \frac{\partial L}{\partial z_{n,m}} \; if \; z_{n,m} > 0 \\ 0 \; otherwise \end{cases} \tag{13}$$

As a result of the above formulation, it is clear to see that after calculating the derivative of the loss function with respect to the output of the network, each layer of neurons needs only to receive an input signal and update their own weights. Each neuron must also scale the loss signal that routes through it to earlier neurons in the network by the weights of that neuron if the neuron outputs a signal greater than 0, and by 0 otherwise.

3.3 Proposed Photonic Neuron for Deep ReLU Networks

Now we return to the motivation of this project: to find a method of on-chip training and inferencing using photonic and electronic components for high-throughput, low-latency computations.

Before we begin discussing the approximate component isomorphisms to desired neuron behaviour, we need to take a slight detour and discuss the behaviour of optical microring resonators (MRRs). MRRs act as wavelength selective filters that can scale the intensity of light at specific, resonant wavelengths, or couple it onto adjacent waveguides whilst not perturbing information on other frequencies (that is, they perform linear processing).

$$\lambda_{resonance} = n_{eff} \cdot \frac{L}{m} \tag{14}$$

Equation 14 details the resonant wavelengths that can be found for a material with n_{eff} effective index, L being the distance an optical signal will travel around a microring resonator, and m being a non-zero positive integer.

Since m is theoretically an infinite set (though is practically limited by the high loss of the waveguide at ultraviolet wavelengths), there is an infinite number of wavelengths that will become resonant when travelling within the MRR. As the distance around the ring is fixed upon construction, ring resonators can be tuned thermally or electrically to change the material's effective index and thus the resonant frequency of the ring.

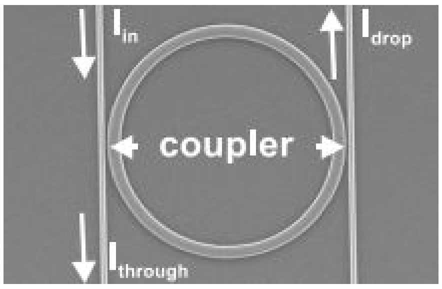


Figure 15: SEM image of a MRR acting as a couple between two waveguides. Taken from [108]

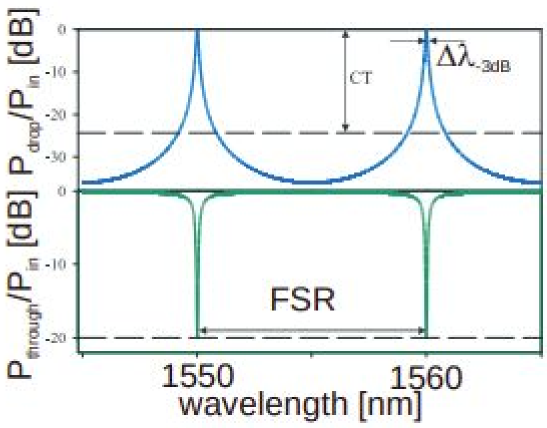


Figure 16: Example of the power in the drop and through port of a loss-less microresonator with F ~ 100 as a function of wavelength. Taken from [108]**.**

It is important that a design of an on-chip photonic neural network incorporate on-chip training to increase the potential use cases and increase industry adoption, and mitigate issues regarding inferencing on resource constrained devices (specifically the low size constraint, as photonic components are inherently larger than electronic components).

An abstract diagram of the entire photonic neuron is provided in Appendix C1, and a complete diagram of the entire photonic neuron is provided in Appendix C2.
The entire photonic neuron design is elaborated upon below.

For the forward propagation of information, we require a neuron design that can:

a) individually weight incoming signals
b) sum the weighted signals
c) add a bias
d) perform the ReLU function on the result
e) propagate this signal forward to later neurons

We will use the design introduced in [92] to form the basis of our feedforward electro-optic neuron, but add circuits for on-chip training and a back-propagated error signal.

The feedforward portion of the neuron will now be discussed in detail for the sake of completeness.To satisfy constraints a) and b), we will use a set of add-drop microring resonators in the form of a silicon-photonic weight bank and a pair of photodetectors.

The proposed method of photonic weighting and summation is shown below, in Figure 16.

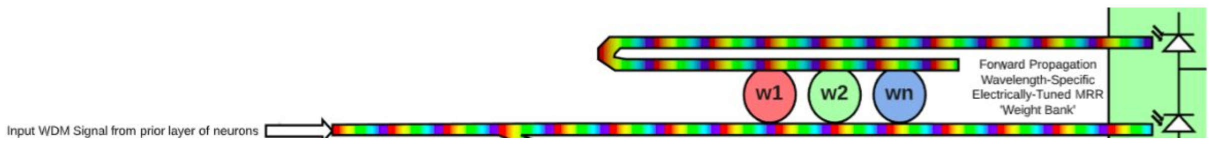


Figure 17: Diagram of weighting and electrical summation of n channels on a wavelength division multiplexed waveguide. Add drop resonators for each channel are used to weight a signal between the negative maximum (wherein the entirety of the channel remains on the input waveguide) and the positive maximum(wherein the entirety of the channel intensity is transferred to the top waveguide). This performs the relative weighting of input signals, which are then summed via incoherent, balanced photodetection.

Assuming good linearity between the incident light intensity and the current produced by the photodetectors, the current i^{+}-i^{-} will be the difference of the average intensity of the signal on each waveguide. As a result, it is equal to the weighted sum of the inputs on each respective channel.

Constraints c,d and e are satisfied jointly ,by using the transfer characteristics of a MRR. When the MRR is tuned away from its resonant frequency, the forward propagating signal's intensity is so small as to be negligible. The current i^{+-}-i^{-} and the bias current are combined then tune the MRR through carrier injection to a value larger than this negligible base intensity; the ReLU function is now open, meaning that the output is now not zero, and signal is of sufficiently large intensity to be considered to be propagating forwards.

While the MRR should, at the time of manufacturing, be designed such that its resonant frequency is as desired, manufacturing defects may mean this is not correct, and so thermo-optic tuning of the MRR can be performed to ensure the required resonance wavelength is tunable.

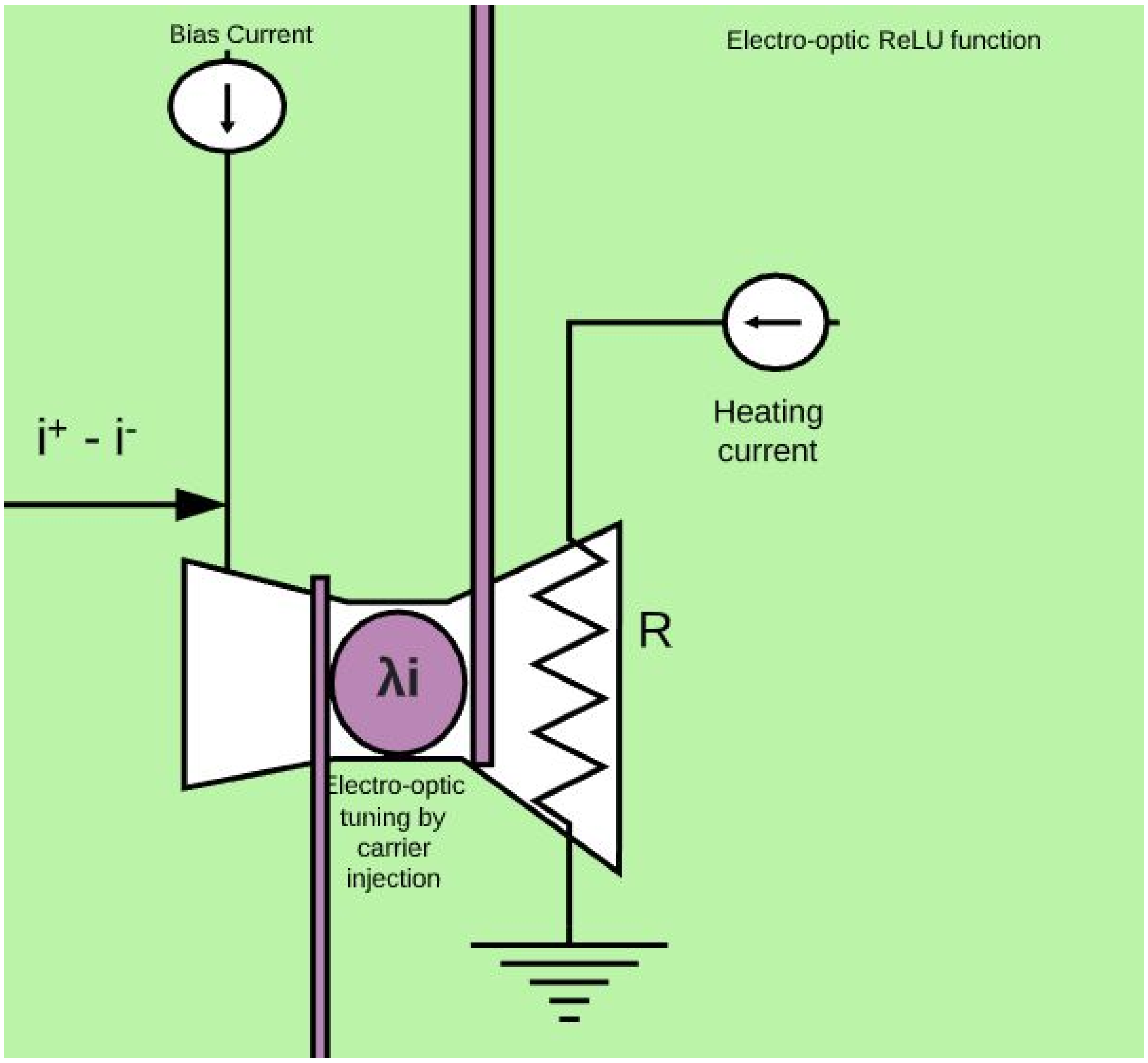

Bias Current
Electro-optic ReLU function
Heating current
$i^+ - i^-$
R
λi
Electro-optic tuning by carrier injection

Figure 18: Electro-optic ReLU function performed by a MRR

A MRR is used to send information to the next layer of neurons, but importantly this could be replaced by an active light source such as an integrated semiconductor laser. For simplicity's sake, given that the chip already requires electro-optically tunable ring resonators, we have chosen an MRR to modulate the signal that is sent to the next layers. This reduces the need for the more advanced manufacturing processes that would be required to create on-chip coherent light sources.

Importantly, the intensity of the signal provided to this forward propagation MRR must be known in advance, to prevent some neurons from sending forth nominally larger or smaller signals than their layer-wide counterparts. This is implemented by a fractional splitter and passive ring resonator to ensure the potential intensity each neuron can tune to is of equal intensity. This is shown in Figure 19.

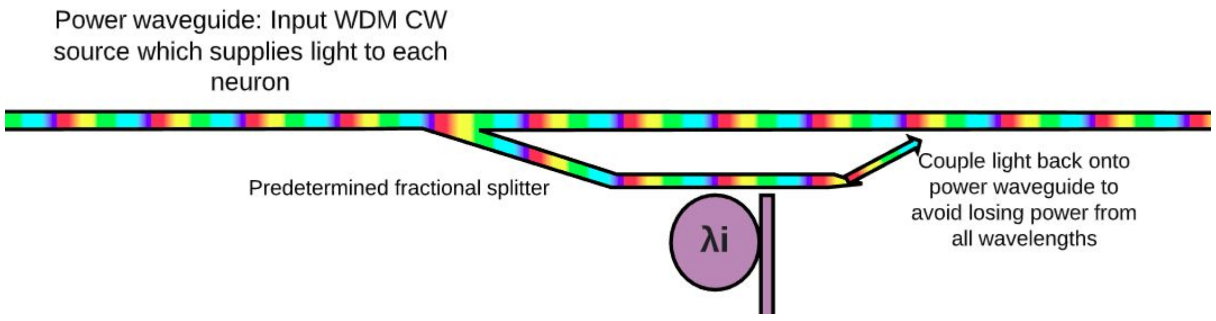

Power waveguide: Input WDM CW source which supplies light to each neuron
Predetermined fractional splitter
λi
Couple light back onto power waveguide to avoid losing power from all wavelengths

Figure 19: Power waveguide, fractional splitter and passive MRR ensure constant, regulated power on a single wavelength to each neuron by design ahead of time. The power waveguide can be attached to a light source on or off chip, removing the need for local light generation. Wavelengths not used by the neuron are coupled back into the power waveguide, to prevent unnecessary loss of optical power.

The single frequency output from a photonic neuron is then combined with signals from other neurons in parallel, to form a WDM signal which can be fed into the next layer of neurons, as shown in Figure 20. Importantly, these signals need not be coherent, as they exist on different frequencies and will not interfere with one-another. By combining these inputs as described, we can then amplify them all at once using only a single optical amplifier if required, and propagate a WDM signal forwards to all the neurons.

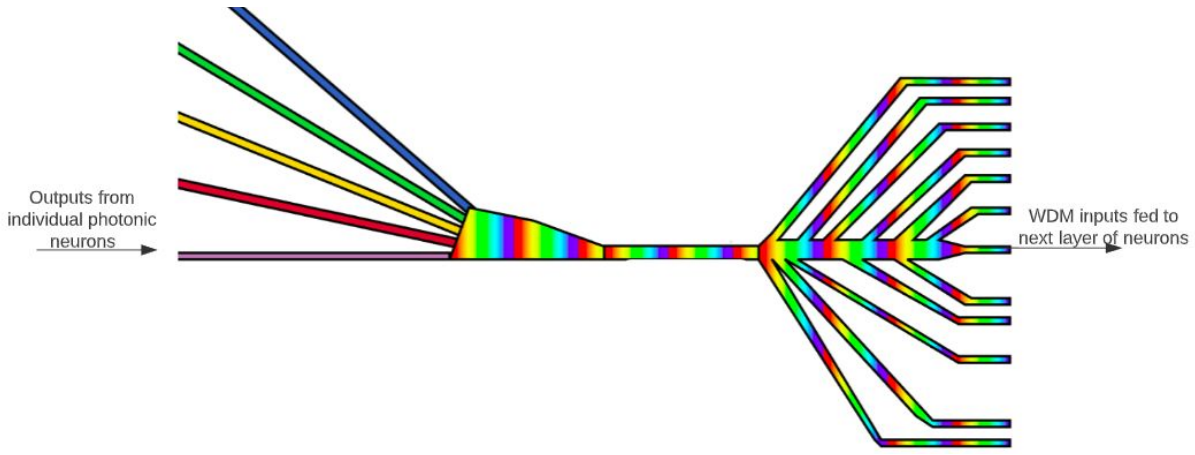

Outputs from individual photonic neurons
WDM inputs fed to next layer of neurons

Figure 20: Diagram of individual neuron outputs, each comprising a single wavelength, being combined and then spatially demultiplexed, to feed WDM signals into the neurons of the next layer.

Provided the above constraints are quite satisfied on-chip with a reasonable signal to noise ratio, such that information of a single bit or more of information can be passed forward to neurons in the next layer reliably, then the construction of a feed-forward photonic neuron has reached its minimum requirements.

To satisfy the additional desired capability of performing on-chip training via stochastic gradient descent utilising back-propagated error derivatives, we require a neuron design that can:

h) Calculate the derivative of the loss function with respect to the network output.
i) Store the incoming forward propagating signals for use in the gradient calculation whenever the ReLU function is greater than 0.
j) Gate the backwards propagating error signal according to whether the ReLU function is greater than 0 or not.
k) Store the incoming loss signal for weight update.
l) Scale the backwards propagating error signal by the weights of the neuron.
m) Update the weights of the neuron according to a weight update rule (for simplicity and efficacy we proposed using Stochastic Gradient Descent)

To satisfy constraint h, which involves calculating the derivative of the loss function with respect to the network output:

For the purposes of this analysis and proposal, we assume that the derivative of the loss function with respect to the network output is available, and design of a system to do so is beyond the scope of the current document. The number of potential loss functions is exorbitant, and provided a single, high-frequency electronic component with clock rates equivalent to the clock rate of information being passed into the network, it would be possible to convert optical information into the electrical domain for more involved calculation before converting this information back into the optical domain to propagate backwards through the network.

For the sake of completeness, however, if we consider a loss function such as a half-mean-squared-error loss function, then the derivative of this loss function is simply equal to the desired output minus the actual output.

Regardless of the method of calculating the derivative of the loss function with respect to the output of the network, this proposal requires a contiguous method of encoding the output of this calculation.

We can encode this optically in a number of ways, and will discuss two here.
We could have two waveguides, one which carries a nominally positive loss value and one for a relatively negative loss value, and have each neuron connect to both of these waveguides to ensure that they receive the proper loss information.

Alternatively, we could use the phase information of light to store the negative and positive values, but we would need a reference wavelength to compare to. We use the power waveguide's phase as the reference, and consider signals which after the loss function interference are 180-degrees out-of-phase with this signal to be positive, and signals in phase with the network output to be negative. Thus, in order to calculate the desired output of the network minus the actual output of the network, we can encode the desired output as being 180-degrees phase delayed from the original, such that if the actual output was greater than the desired output, signal remains at the original phase of the network, and if the actual output of the network on some channel was less the desired output, the result would be some intensity greater than zero on a phase of 180 degrees.

Then, we need to pass this information, and its relative phase back to the last layer of neurons. Since the relative phase needs to be known, we can use the power waveguide again, and route this phase information along a second waveguide that travels to each neuron equidistant with the error signal waveguide.

To satisfy constraint j, which involves gating the backwards propagating error signal according to whether the neuron's output is greater than 0 or not:

In order to gate the incoming error derivative signal, we propose using an electro-optically tunable ring resonator, which has resonant peaks at all of the WDM channels. This component could be replaced by any other band-pass, tunable optical filter.

When the forward propagating signal is greater than the nominated amount, and the output of the neuron is thus assumed to be greater than zero, a fraction of this light will be coupled onto another waveguide and then detected electro-optically. If the intensity of this light is greater than a predetermined threshold corresponding to the nominated 'zero' threshold, then this neuron is taken to be propagating signal forwards, and as such should allow all backwards error signals to propagate back across the neuron. If the intensity of this light is less than the 'zero' threshold, then this filter should close and prevent all error signals from propagating backwards across the neuron, or being sampled by the neuron.

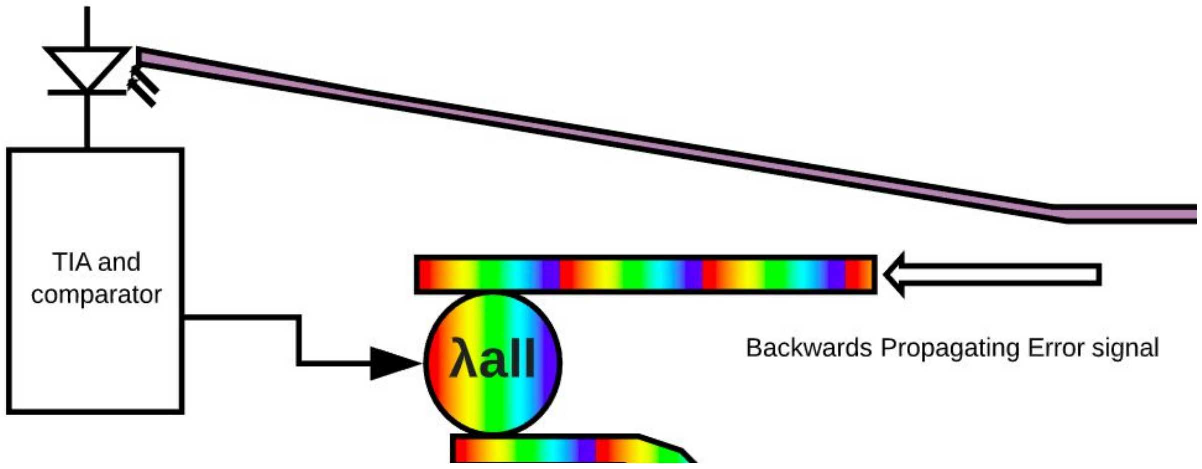


Figure 21: Gating optical filter for back propagated error selection via forward propagating optical signal thresholding.

To satisfy constraint k, which involves storing the incoming loss signal for weight update:

We can store the incoming loss signal, for later use in weight updates, by using an analog integrator circuit attached to a pair of balanced photodetectors. Since we proposed encoding negative loss values as a phase difference relative to the power waveguide, we need a phase aware method of detection of intensity relative to the power waveguide.

It is proposed to siphon a signal, using a fractional splitter with a predetermined ratio, off of the power waveguide that has travelled alongside the loss waveguide, and further split this signal 50:50 onto two separate waveguides. The first will remain relatively unchanged, and the second waveguide will be delayed such that its phase is 180° relative to the first waveguide. We will also split the local loss signal 50:50 onto these two waveguides. Then, the light in these two waveguides will be incident upon two photodetectors which sum the available intensities.

The relative difference in the currents created by these photodetectors will be, provided relatively linear photodetectors are used, proportional to the sign and magnitude of the derivative of the loss function relative to this neuron's output.

We can, using a high-bandwidth analogue integrator, store this current over time, and use this stored value to sample from later on, to update weights and biases.

For this to work it is important that the power waveguide moving backwards is routed such that it travels the exact same distance to each neuron as the loss signal has, ensuring that no relative phase change owing to different distances travelled occurs.

To satisfy constraint I, which involves scaling the backwards propagating error signal by the weights of the neuron:

Now, we have the derivative of the loss function relative to the local neuron. We have seen above that in a deep ReLU network, each neuron needs to scale the loss signal it feeds to prior neurons by the weights of each individual input it received.

In order to scale the loss function correctly, taking into account that a loss signal 180° out of phase with the power waveguide is considered 'negative', we will scale the loss signal by the forward propagating weights using MRRs, such that a weight of 1 corresponds to the signal being completely unchanged, and a weight of -1 corresponds to the MRR for that wavelength being tuned to resonance, such that the wavelength is coupled onto a separate waveguide and phase delayed, before combining back with the original local loss signal. As such, a weight of 0 would then correspond to half of the wavelength being coupled onto the other waveguide, phase delayed by 180 degrees and then recombined with the other half of the wavelength's intensity, which would then destructively interfere and leave no power on that wavelength propagating backwards from this neuron.

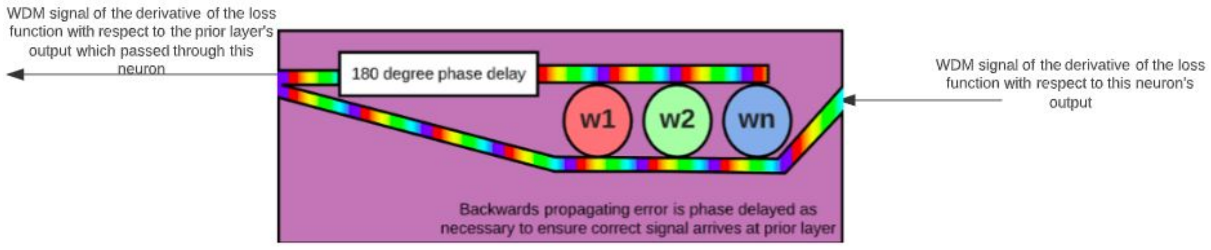

WDM signal of the derivative of the loss function with respect to the prior layer's output which passed through this neuron
180 degree phase delay
w1
w2
wn
WDM signal of the derivative of the loss function with respect to this neuron's output
Backwards propagating error is phase delayed as necessary to ensure correct signal arrives at prior layer

Figure 22: Diagram of the phase-sensitive scaling of the backward propagating derivative of the loss function with respect to the current neuron's output.

After each neuron scales the backward propagating loss signal, these signals then need to be recombined, such that the signal from each neuron travels an equal distance and the signals interfere constructively and destructively as required, before then being split equally to connect to each of the neurons in the prior layer.

To satisfy constraint m, which involves updating the weights of the neuron according to the SGD update rule:

Once the derivative of the loss function with respect to the local neuron has been stored, as outlined when satisfying constraint k, in the integrator's voltage, we can sample from this voltage, and from the stored value of the inputs into the neuron and uses these values to update the weights of the neuron. It is proposed to do this using a RF multiplier, such that the weight update rule for SGD is satisfied. Specifically, in order to perform gradient descent, we need to change the existing weight by the negative of the learning rate *n* multiplied by the derivative of the loss function with respect to the weight at that neuron.

A simple example circuit, modelled in Simulink, of the proposed analog weight update is shown in Appendix B, and the individual subcircuits are expanded upon in Appendix B Figures B2-B7, with sample simulation results shown in B8. The simulated results shown in diagram B8 demonstrates that the SGD weight update rule can be performed with negligible error on simple, inexpensive analog electric components. As such, analogue electronic circuits which can perform the SGD weight update rule are able to be integrated, and this makes monolithic integration of photonic and electronic components more viable.

Using a set of op-amps and an RF-multiplier, with a sample and hold (SH) circuit to store the weight of the neuron as it is being updated, we can update the local weight of the neuron, which is stored as a voltage on a second SH circuit.

Then, an external clock signal needs to reset all of the integrators attached to the neuron, to ensure the next weight update is mathematically valid. This weight-update needs to occur when there is no signal propagating through the network, to correctly perform synchronous SGD and avoid transitive weight positions contributing to the loss signal accumulated by the neuron.

Importantly, a weight update does not need to occur after each sample fed to the network, as the integrators attached to the storage circuits can store the cumulative signal observed over multiple examples, allowing 'batching' of inputs to occur as desired.

A model of the circuits required for this weight-update system has been developed, and verified by simulation.

This stored weight voltage can be used to thermally tune the MRRs using either thermo-optic or electro-optic tuning. As this is a well explored area of research, it is left as a choice to the designers at time of manufacturing as to the type of tuning and circuits used for this particular task.

To satisfy constraint i, which involves storing the incoming forward propagating signals for use in the gradient calculation whenever the ReLU function is greater than zero:

In order to correctly calculate the weight update function for a neuron, as seen above, we need to store the incoming inputs. We can choose to do this by way of a passive MRR and photodetector attached to an integrator circuit, which stores the inputs to the neuron on each individual wavelength. In order to avoid storing values in these integrator circuits which did not cause the neuron to output a signal and so did not contribute to the loss of the network with respect to an example, we need to attach a photodetector, TIA and comparator to the output waveguide of the neuron, so that the storage of input signals to the neuron can be gated. This is shown in Figure 23, wherein the all or none passing microring resonator denoted λall is electro-optically tuned via a comparator attached to the forward propagating output of the neuron.

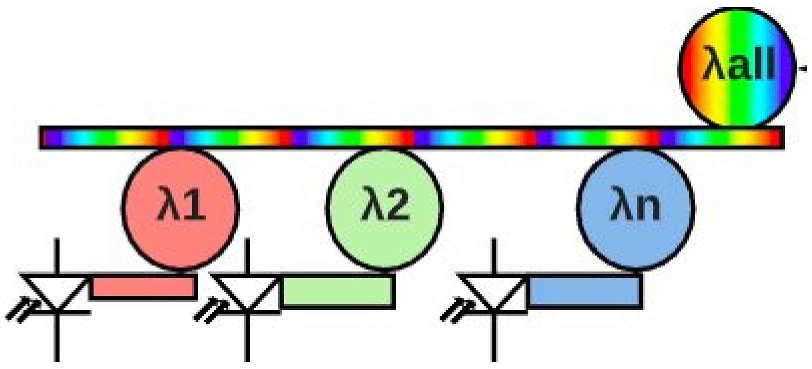
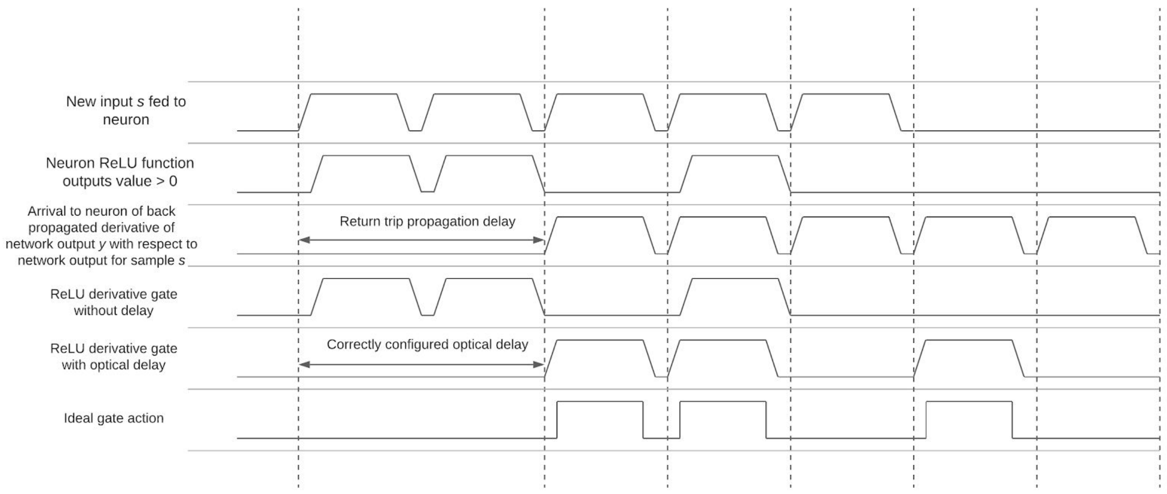


Figure 23: Passive MRRs and photodetectors to sample inputs coming into the neuron.

As a further addition, which will improve the throughput of a connected network of these photonic neurons, we suggest the addition of a variable time delay line attached to the output of each neuron, such that the signal fed to the backwards propagating error gate, and thus the gate's functioning, is synchronised with the actual backward propagating error signal.

If the size of the neural network being created is fixed and known ahead of time, then this delay can be implemented simply by a fixed-length waveguide. If the chip, as is probable, will be used to train more than one neural network, such that the distance of a given neuron from the end of the network is not known ahead of time, then this needs to be a variable delay component. Each neuron only outputs a single wavelength, and variable length optical delay components have been researched and implemented in industry widespread enough that this should not be a serious hindrance, although the size of this variable length optical delay component will need to be considered in advance.

Without a delay line to synchronise the action of the gate with the backpropagating signal temporally, there is still a way to train the proposed photonic neural network on-chip, but it means that only a single example can be present in the network at the same time, with a gap in-between samples equivalent to the latency of a signal travelling all the way forward through the network and back, such that the first layer of neurons accumulate loss signals for the same period of time as later layers of neurons.



New input *s* fed to neuron
Neuron ReLU function outputs value > 0
Arrival to neuron of back propagated derivative of network output *y* with respect to network output for sample *s*
Return trip propagation delay
ReLU derivative gate without delay
ReLU derivative gate with optical delay
Correctly configured optical delay
Ideal gate action

Figure 24: Back propagating derivative gate timing diagram. For this simple example, the ReLU derivative gate without delay acts ahead of time of the returning back propagated derivative signals, and as a result does not correctly gate the derivatives of the first and third sample, whereas the ReLU derivative gate with delay acts as desired.

Without this delay between samples, the gates at each neuron could open or close for a new sample, and the neuron may thus incorrectly accumulate loss signals that it did not contribute to, or it may not accumulate loss signal despite contributing to the network output, so that the neurons in the network no longer adhere to the rules of stochastic gradient descent, and as a result a degradation in performance, and potential failure to converge, would be observed. A simple example of this erroneous behaviour is shown in Figure 24.

Alternatively, with a delay line at each neuron, we have no need for a gap in-between samples, as each neuron temporally aligns the backpropagating signal with the signal it propagates forward for each example, such that feeding a new example immediately after another example to the network will not result in improper gradient accumulation or backpropagating loss signal gating, aside from the propagation delay of the circuit at each neuron which gates the backpropagating error signal.

Thus, with all of the constraints for a feedforward optical neural network with on-chip stochastic gradient update of weights based upon physically back propagated error signals satisfied on paper, and with neural networks able to effectively learn with 1 bit or more of information, it is entirely plausible that the described and laid out architecture of a photonic neuron is useful.

All of the components required in the photonic neuron proposed within this document have been individually demonstrated in both industry and research, and effective training of neural networks on quantized and even binary information has been demonstrated and replicated, so the potential barriers to a photoelectric neuron are rapidly eroding.

Assuming that the recent industry and academic focus on photonics leads to both advances in monolithic manufacturing of electronic and photonic components, and a reduction of the costs associated with creating an opto-electronic chip mean that the network proposed above could be manufactured and experimentally verified in the coming years.

3.4 Analysis of a photonic neural network comprising the prescribed WDM photonic neuron

The photonic neuron described above has already considered a number of network level constraints, such as scaling the backpropagating error signals locally and encoding the sign of loss information into the relative phase of the signal.

In this section we will perform some back-of-the-envelope calculations to reason about the size of an on-chip photonic neural network. In order for these photonic neural networks to run at their optimal throughput and not be throttled by digital electronic calculations, it is important that as many computational components are running at photonic and analogue electronic rates. As a result, no parallelism or physical re-use of individual components should occur in the processing of a single instance of information, to prevent the switching of weights and biases of an individual neuron being throttled by digital electronic components and ADCs. If this were the case, it would likely be easier to keep the information in the electric domain, and transform and compute with it using more traditional digital means.

Thus, to avoid reusing components, we require a physical analog of each neuron in the network being calculated. To estimate the on-chip space requirements of a single neuron, we shall evaluate the number of each component and their relative size.

To scale the forward propagating signals by individual weights, we require *n* individual actively tuned microring resonators for *n* WDM channels. To scale the backward propagating error, we require *n* individual actively tuned microring resonators, which can be considered clones of the forward weighting MRRs, as they should have identical weights. We also require *n* individual passive MRRs to sample the intensity of inputs to the neuron on each WDM channel. There are 4 additional microring resonators at each neuron, one which couples power from the power waveguide, one which gates the sampling of neuron inputs, one which gates the back-propagating error signals according to the output of the neuron and one which modulates the power that is propagated forwards. Thus, for *n* inputs, we require 3*n* + 4 MRRs.

The design requires *n* photodetectors to sample the intensity of inputs to the neuron. It needs 2 photodetectors to perform the balanced photodetection required to implement the weighting of photonic information. It also needs one photodetector to gate all of the input sampling storage circuits for all the neuron's input WDM signals, and another photodetector for the gating circuit of the back propagating error signals. Two additional photodetectors are needed for the balanced photodetection of the local error derivative at the neuron, such that the sign and magnitude of the error signal can be measured and used to update neuron weights. Thus, for *n* inputs, we require *n*+6 photodetectors.

Each neuron also requires a variable optical delay line, to synchronise the gating of backward propagating error signals with the back propagating error signal itself.

Lastly, and perhaps least significantly in terms of on-chip area required, each neuron will require some active analogue electronic components. The weight update circuit requires an integrating op-amp for each neuron input WDM channel sampling circuit, and an integrating circuit to store the local error derivative over multiple.

Table 1: Components required to build the proposed photonic neuron and the on-chip area consumption of these components

Component	Individual Size (μm^2)	Number of components for *n* channels	Total size for *n* WDM channels (μm^2)	Total size for 100 WDM channels (μm^2)	Cumulative chip area required for 100 WDM channels(μm^2) (assuming 40% extra

					space required for placement of components)
MRRs	20	3*n* + 4	60*n* + 80	6080	8512
Photodetectors	1	*n* + 6	*n* + 6	106	8660
Weight update circuit, TIAs and comparators	30*	1	30*	30*	8705

*given that the size of electrical components are orders of magnitude smaller than photonic components, we assume the weight update circuit is a constant size for any number of channels, as the total on-chip area will be dominated by the number of microring resonators required

For a simple neural network with 784 inputs (for a 28x28 image), a hidden layer of 784 neurons and 10 output neurons (a common network configuration used for the MNIST handwritten digit recognition task), we thus need a minimum chip space of 8705 μm^2 x (784+10) = 6.91 mm^2.

This is a non-trivial size for an integrated chip, and this large size may make manufacturing substantially more likely to result in a flawed chip. Importantly, this is also a trivially small neural network that would be easily trained by a modern computer on a CPU alone, and, perhaps more importantly, the above estimated area of a neuron doesn't take into account the relatively enormous size of the variable optical delay line at each neuron.

The size of the variable optical delay line required at each neuron can be mitigated somewhat, by having a shared optical delay line at each neuron layer. Each neuron in a layer outputs a unique wavelength, and each neuron in a layer has an equivalent distance to the output of the network, and so a single shared optical delay line at each layer of neurons is a potentially space saving and desirable optimisation.

The above size does not take into account peripheral electronics required for this photonic neural network, such as an active light source and the electric and photonic components required to perform the calculation of the derivative of the loss function with respect to the network outputs.

However, by using physically analog neurons, we come to arguably the most important design factor in modern deep learning focussed hardware accelerators: computational throughput. With physically parallel neurons in a layer, and the ability to feed multiple examples to the network at any one time, the throughput of a photonic neural network comprising the above photonic neuron is enormous.

According to [85], the reading of weights as light propagates across a MRR takes a few tens of picoseconds to occur, the intra-layer combination of WDM signals takes only a few picoseconds to occur, and the ReLU activation circuit can operate at tens of picoseconds timescales also. Thus, for a forward propagating signal, each layer is expected to require less than 100ps to read an input and pass a calculated output to the next layer.

The backwards propagating signals from layer to layer remain in the optical domain and are never converted back into electrical signals. As such, the time required for a signal to propagate backwards across an individual layer could be as low as 50ps.

Given these rough values, for a 10-layer neural network, an example could propagate forwards throughout the entire network in less than 1 ns, and could then propagate backwards across each neuron in only an additional 500 ps.

Given that the active analog electronics used to sample inputs will not react instantaneously, we also need to hold the same signal for some time. With reasonably fast analogue electronics (we shall assume 5 GHz speeds are achievable), we need to hold each sample for a minimum of 200 ps. Thus, the total time for a single example to propagate forwards and back over a 10-layer circuit is 1.7 ns. Assuming also that some 300 ps dead time is required between examples to allow functions to settle back into a base state, and that some 500ps is required for the loss function application at the network outputs, we have that an example can propagate backwards and forwards over the entire 10-layer neural network in 2.5 ns.

Some additional time is required during training, to update the weights of each neuron while no sample is being fed in. A reasonable assumption is 10 ns for these weight updates, as RF multiplication and updating values stored in the weighting sample and hold circuits is required. This weight update can also be performed in parallel across the entirety of the chip. Thus, assuming the sample and hold circuit can hold a weight for a minimum of 100 samples, which would take 250 ns, we have an amortized weight update cost per example of 40ps. Thus the total amortized time required per sample fed into the network is 2.6 ns.

With training speeds like this, a monolithically integrated 10-layer photonic neural network could train on $3.8x10^8$ samples per second. This can be further improved, however, since we can interleave examples, such that the network doesn't need to wait until the error derivatives from the last sample have propagated back to the first layer before feeding in the next example.

Considering again that the input sampling circuits require that a sample be held constant for 200 ps to sample, and that a dead time of 300 ps is required between samples, we can feed in a new example to the network every 500 ps. As a result of this, it would take at most 50 ns to feed in 100 samples, and an additional 2.5 ns for the error derivatives from the last example fed in to backpropagate fully across the network. Addition of the 10 ns required for the weight updates to this results in an amortized time per sample for an interleaved network of 625 ps per sample.

With this increased training speed, a monolithically integrated 10 layer photonic neural network could train on 1.6 giga samples per second.

The above calculations all hinge on the premise that a 10-layer neural network can be realised and trained end-to-end without having to shuffle intermediate results into electronic memory.

If the on-chip photonic neural network lacks the WDM channels required to calculate all of the inputs at each neuron as required, then training will be substantially slower, as intermediate results would need to be stored in memory while slices of the network outputs are computed, concatenated and transformed once again.

As such, the two largest throttles to the throughput of this network will be the fan-in of each neuron and the number of neurons available in the network. Since each neuron is essentially homogeneous, the depth of the network can be (somewhat) arbitrarily expanded by routing one chip's outputs off-chip, where it would be actively boosted, and then coupled onto subsequent chips for further calculations. As such, each chip should likely only contain a single layer, and the width or number of parallel neurons in this layer should be maximised, to allow the most smaller networks to exist within this maximal outer network.

Thus, the true largest design flaw of a network of these neurons is the fan-in of each individual neuron. Independent WDM control of 60 channels or more has been demonstrated, but this pales in comparison to the fan-in of digital and biological neural networks.

3.5 What are the roadblocks?

The above outlined photonic neuron works in principle, and lengthy consideration has been given in this design into exactly how a network of these neurons would be created and function.

Numerous shortcomings still exist, however, and I will attempt to address all that exist in the pages that follow.

The above photonic neuron requires monolithic integration of photonic and electronic components, to avoid enormous on-chip/off-chip coupling losses for each PD and MRR.

Additionally, given the relatively large size of photonic components as compared to electronic components, it is crucial that efforts to monolithically integrate the components are integrated, as this will prevent the already potentially prohibitively large neuron from requiring even larger on-chip/off-chip coupling components and electronics, further inflating the size.

Monolithic integration of these components is difficult to say the least, and would require investment of upwards of millions of dollars in trial and error to satisfactorily create, iterate upon and deliver the proposed photonic neural network. Testing and verifying the individual performance of components such as photodetectors, active microring resonators and the weight update circuitry, whilst also ensuring minimal on-chip crosstalk between components such that they function appropriately together is not a trivial task. Radio feedback and interference may become an issue without careful consideration of shielding and other optimisation techniques.

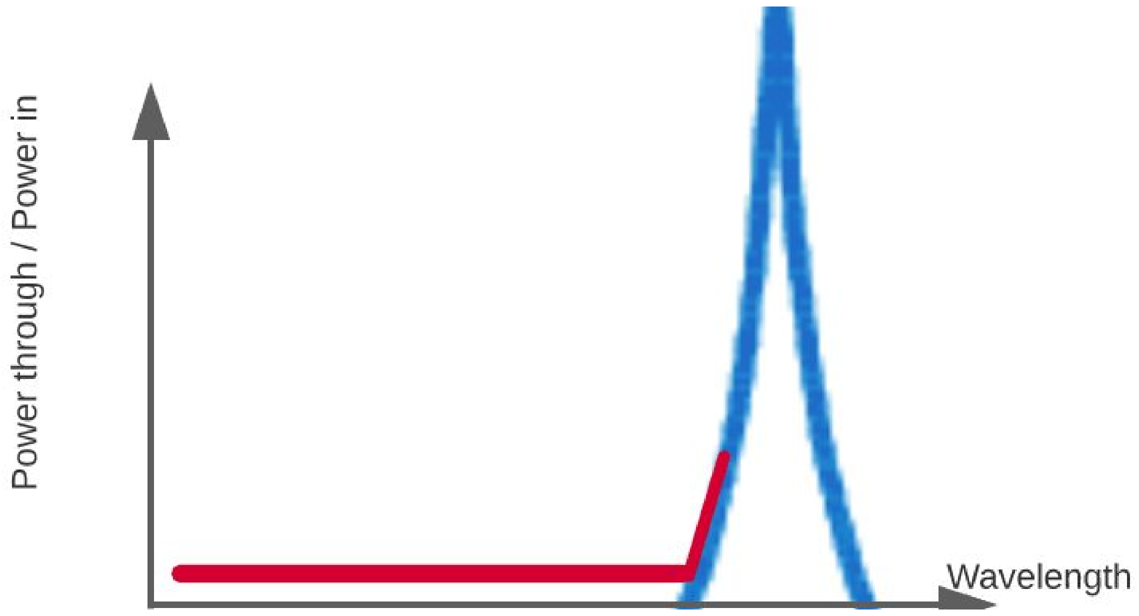


Figure 25: Desired MRR ReLU functionality. The blue curve is the tip of a MRR power through/power in profile, whilst the red curve is the desired ReLU functionality.

The photonic neuron design also assumes, by nature of the creation of a ReLU function using the slope of the MRR and the on/off characteristics of a PN junction, that a linear MRR-current response curve is achieved, such that the diagram in Figure 24 is attainable. The clear on-off power may be achievable through a PN junction, but the linearity of the MRR response (a resonance is usually Lorentzian in shape, with curved sides, which can be approximated to be linear for small enough changes in intensity) when it is being electro-optically tuned will have to be investigated. Also of importance to note, is that calculations are performed during the design process such that, if a linear slope is achieved, each MRR is not tuned beyond the point of highest resonance. This will ensure that for monotonically increasing input powers, a neuron's output will also be monotonically increasing (provided the input power is greater than the nominal zero and thus enough to break from the ReLU threshold). By nature of this assumption, it is not a calculation that can be performed ahead of time, as it requires knowledge of the maximum output power of any neuron in a preceding layer, and the number of neurons that signal fans out to, and the number of neurons which feed a signal into a specific neuron, which are tied to manufacturing decisions not yet completed.

The photonic neuron outlined in this document also has no switching or reused components by design. This is not a design flaw per se, and instead a compromise, as the architect sought to increase the overall throughput of the system by removing the requirement to switch components, however it comes at the cost of an increased number of components, and crucially, the inability to reuse components. This may hinder use cases somewhat, as it means that a neuron created with, for example, 20 inputs will never be able to utilise more than 20 inputs at a time. This may be a common tradeoff considered when creating analog components, but an analog circuit such as the one outlined above will not be able to dynamically adjust and reconfigure the same as a digital circuit would.

The power consumption of the number of active analog components may also end up being non-trivial. Using the circuits developed to perform weight updates, there are a number of op-amp based integrators, sample and hold circuits, and one RF multiplier for each WDM input channel at each neuron. A system using chalcogenide or some other passive optical component to set weights may ultimately be more performant than the design outlined above, as it reduces the amount of active electronics and thus power drawn, but this design choice would further complicate the monolithic integration of components.

3D integration of waveguides and passive and active photonic components may become a necessity to reduce the on-chip space required by a network of photonic neurons, but comes at the cost of increased manufacturing complexity and non-trivial computation required to calculate optimal routing pathways. However, since light propagates almost losslessly in a silicon waveguide, little heat is created, and so only the active electrical components need be at or near the chip surface for heat sinking, as waveguides will not generate excess heat in comparison. As such, further investigation into three dimensional waveguides for transmitting photonic information on chip is an attractive avenue of further research.

This raises another pressing issue. Thermal behaviour of the combined photoelectric neuron will have to be studied and compensated for, such that the chip is able to function in real world temperatures ranging from -40 to 80 degrees Celsius (generally the design constraints for digital chips). If these chips are only intended to be used in a server, then it is possible that this range of operating temperatures could be refined further, but at the bare minimum the chip would need to be certified to operate as expected from 20 to 55 ^{o}C.

Assuming good linearity and control of the optical components, there is an additional error introduced by the gating of the sampling of neuron inputs: since this circuit will not operate instantaneously, it stands to reason that the gating of the inputs will lag changes in inputs, and as such the neuron input intensity value stored over time may not be a good representation of the actual inputs the neuron has seen. This effect will not be present when subsequent samples fed to the network produce the same neuron response of greater than zero or zero, but whenever two subsequent samples cause the neuron output to switch from 0 to higher or vice versa, the gate delay error will come into effect. By placing temporal gaps between the examples fed into the network with gap time equal to the response time of the input gating circuitry, we can mitigate this error, albeit at the cost of network throughput.

Sharing a variable optical delay line at each layer of neurons in the network will reduce the volume required for each neuron, but comes at the cost of increased neuron and layer complexity, as frequency specific filters will be required to couple signals out of the shared variable optical delay line. This may also be a non-trivial component to manufacture and optimise and numerical optimisation of such a component, in parallel with the rest of the neuron and chip requirements, on a digital computer will take considerable time.

Another issue that arises upon examining this neuron and subsequent network design is the encoding of loss information onto the phase of the back propagating error signals. If fine-grained control of this phase information is found not to be available across cascading neurons, then other methods of encoding the sign of the loss signal will have to be employed. Specifically, with two separate waveguides, one for positive derivative signals and one for negative derivative signals, control of the phase information is not necessary, but a different method of scaling backward propagating signals will be necessary, as a clone of the forward weighting and a phase delay will no longer be a viable scaling and multiplication method.

4. Conclusions and future work

Over the course of this project, an extensive, thorough literature review was constructed, which served as the author's introduction to the fields of photonics and neuromorphic computing. Once complete, a thorough proposal for a physically analog photonic ReLU neuron that can operate at gigasamples throughputs was constructed, and an analysis of the potential performance of a network of these neurons was reasoned about. The on-chip space required to house these photonic neurons is gargantuan in comparison to digital electronics, but the increases in throughput, latency and energy efficiency likely more than compensate for the increased size.

Server side computing, wherein energy consumption, bandwidth and latency dominate the lesser constraint of area, is especially suited to dedicated photonics coprocessors, such as the WDM photonic neuron proposed in this document, and the HolyLight photonics coprocessors[20].

Despite initial plans, an examination of the performance of one of these neurons with respect to noise was not performed. Without more pinned-down component specifications, manufacturing capabilities, and power budgets, there were simply too many degrees of freedom and as a result any signal to noise ratio estimated would be too sensitive to change to be worth calculating. Additionally, for an automated solver to estimate the signal to noise ratio of a neuron would require the existence of a solver that can model both analogue electronics and photonics natively, and concurrently. To the author's knowledge, this software does not yet exist.

Future works will seek to focus on a more modular design, wherein a single layer of neurons exists only on one chip, and chip-to-chip coupling is utilised to build out a more expansive photonic neural network. By coupling multiple chips together and limiting each chip to a single fully connected layer of neurons, the size of flawless silicon die sizes required to be manufactured decreases, which decreases the cost of manufacturing. Additionally, optical amplifiers could be easily placed between layers of neurons, enhancing the potential depth of a cascaded network, and shared variable optical delay lines could be constructed with wavelength specific filters and off-chip delay lines, resulting in a smaller form factor for the design.

Additionally, while the ReLU function is widespread and simple to work with, creating a different, more flexible non-linear activation function and respective derivative function would allow for more network expressivity and may increase the effectiveness of the photonic architectures. In particular, sinusoidal activation functions may be both easy to construct in analogue electronics and provide useful for photonic implementation of recent, deep fully connected neural networks[109].

A link to a video explaining this project can be found here:
https://youtu.be/OoOzpuUT3sg

A link to a Github repository containing both this report and the Simulink model used to simulate the operation of the weight-update circuit is available here:
https://github.com/njgre6/FYP-Photonics-Neuromorphic-Optics

5. References

[1] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, “Analyzing and Improving the Image Quality of StyleGAN,” *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 [Online]. Available: http://dx.doi.org/10.1109/cvpr42600.2020.00813

[2] H. Zhou, S. Hadap, K. Sunkavalli, and D. Jacobs, “Deep Single-Image Portrait Relighting,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/iccv.2019.00729

[3] J. Thies, M. Elgharib, A. Tewari, C. Theobalt, and M. Nießner, “Neural Voice Puppetry: Audio-Driven Facial Reenactment,” *Computer Vision – ECCV 2020*. pp. 716–731, 2020 [Online]. Available: http://dx.doi.org/10.1007/978-3-030-58517-4_42

[4] Y. Nirkin, Y. Keller, and T. Hassner, “FSGAN: Subject Agnostic Face Swapping and Reenactment,” *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/iccv.2019.00728

[5] L. Shen, L. R. Margolies, J. H. Rothstein, E. Fluder, R. McBride, and W. Sieh, “Deep Learning to Improve Breast Cancer Detection on Screening Mammography,” *Sci. Rep.*, vol. 9, no. 1, p. 12495, Aug. 2019, doi: 10.1038/s41598-019-48995-4. [Online]. Available: http://dx.doi.org/10.1038/s41598-019-48995-4

[6] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/TPAMI.2016.2644615. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2644615

[7] A. W. Senior *et al.*, “Improved protein structure prediction using potentials from deep learning,” *Nature*, vol. 577, no. 7792, pp. 706–710, Jan. 2020, doi: 10.1038/s41586-019-1923-7. [Online]. Available: http://dx.doi.org/10.1038/s41586-019-1923-7

[8] K. Hornik, “Approximation capabilities of multilayer feedforward networks,” *Neural Networks*, vol. 4, no. 2. pp. 251–257, 1991 [Online]. Available: http://dx.doi.org/10.1016/0893-6080(91)90009-t

[9] International Business Machines Corporation. Research Division and J. Backus, *“Can Programming be Liberated from the Von Neumann Style? a Functional Style and Its Algebra of Programs.”* 1978 [Online]. Available: https://books.google.com/books/about/Can_Programming_be_Liberated_from_the_Vo.html?hl=&id=QO-0XwAACAAJ

[10] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*. Elsevier, 2012 [Online]. Available: https://books.google.com/books/about/Computer_Architecture.html?hl=&id=v3-1hVwHnHwC

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *Learning Internal Representations by Error Propagation*. 1985 [Online]. Available: https://books.google.com/books/about/Learning_Internal_Representations_by_Err.html?hl=&id=Ff9iHAAACAAJ

[12] E. Strubell, A. Ganesh, and A. McCallum, “Energy and Policy Considerations for Deep Learning in NLP,” *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019 [Online]. Available: http://dx.doi.org/10.18653/v1/p19-1355

[13] D. Amodei, D. Hernandez, G. Sastry, J. Clark, G. Brockman, and I. Sutskever, “AI and Compute,” 16-May-2018. [Online]. Available: https://openai.com/blog/ai-and-compute/. [Accessed: 15-Oct-2020]

[14] Q. Xu, S. Manipatruni, B. Schmidt, J. Shakya, and M. Lipson, “12.5 Gbit/s carrier-injection-based silicon micro-ring silicon modulators,” *Opt. Express*, vol. 15, no. 2, pp. 430–436, Jan. 2007, doi: 10.1364/oe.15.000430. [Online]. Available:

http://dx.doi.org/10.1364/oe.15.000430
[15] S. Ambrogio *et al.*, “Equivalent-accuracy accelerated neural-network training using analogue memory,” *Nature*, vol. 558, no. 7708, pp. 60–67, Jun. 2018, doi: 10.1038/s41586-018-0180-5. [Online]. Available: http://dx.doi.org/10.1038/s41586-018-0180-5
[16] B. Zhuang, C. Shen, M. Tan, L. Liu, and I. Reid, “Structured Binary Neural Networks for Accurate Image Classification and Semantic Segmentation,” *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/cvpr.2019.00050
[17] C. Huang *et al.*, “Demonstration of scalable microring weight bank control for large-scale photonic integrated circuits,” *APL Photonics*, vol. 5, no. 4. p. 040803, 2020 [Online]. Available: http://dx.doi.org/10.1063/1.5144121
[18] M. D. McDonnell and L. M. Ward, “The benefits of noise in neural systems: bridging theory and experiment,” *Nat. Rev. Neurosci.*, vol. 12, no. 7, pp. 415–426, Jun. 2011, doi: 10.1038/nrn3061. [Online]. Available: http://dx.doi.org/10.1038/nrn3061
[19] Y. Yamada, M. Iwamura, T. Akiba, and K. Kise, “Shakedrop Regularization for Deep Residual Learning,” *IEEE Access*, vol. 7. pp. 186126–186136, 2019 [Online]. Available: http://dx.doi.org/10.1109/access.2019.2960566
[20] W. Liu, W. Liu, Y. Ye, Q. Lou, Y. Xie, and L. Jiang, “HolyLight: A Nanophotonic Accelerator for Deep Learning in Data Centers,” *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2019 [Online]. Available: http://dx.doi.org/10.23919/date.2019.8715195
[21] A. L. Hodgkin and A. F. Huxley, “A quantitative description of membrane current and its application to conduction and excitation in nerve. 1952,” *Bull. Math. Biol.*, vol. 52, no. 1–2, pp. 25–71; discussion 5–23, 1990, doi: 10.1007/bf02459568. [Online]. Available: http://dx.doi.org/10.1007/bf02459568
[22] E. M. Izhikevich, “Which Model to Use for Cortical Spiking Neurons?,” *IEEE Transactions on Neural Networks*, vol. 15, no. 5. pp. 1063–1070, 2004 [Online]. Available: http://dx.doi.org/10.1109/tnn.2004.832719
[23] W. Gerstner and W. M. Kistler, *Spiking Neuron Models: Single Neurons, Populations, Plasticity*. Cambridge University Press, 2002 [Online]. Available: https://books.google.com/books/about/Spiking_Neuron_Models.html?hl=&id=Rs4oc7HfxIUC
[24] E. M. Izhikevich, “Simple model of spiking neurons,” *IEEE Transactions on Neural Networks*, vol. 14, no. 6. pp. 1569–1572, 2003 [Online]. Available: http://dx.doi.org/10.1109/tnn.2003.820440
[25] H. Markram and M. Tsodyks, “Redistribution of synaptic efficacy between neocortical pyramidal neurons,” *Nature*, vol. 382, no. 6594, pp. 807–810, Aug. 1996, doi: 10.1038/382807a0. [Online]. Available: http://dx.doi.org/10.1038/382807a0
[26] H. Markram, J. Lübke, M. Frotscher, and B. Sakmann, “Regulation of synaptic efficacy by coincidence of postsynaptic APs and EPSPs,” *Science*, vol. 275, no. 5297, pp. 213–215, Jan. 1997, doi: 10.1126/science.275.5297.213. [Online]. Available: http://dx.doi.org/10.1126/science.275.5297.213
[27] G. Q. Bi and M. M. Poo, “Synaptic modifications in cultured hippocampal neurons: dependence on spike timing, synaptic strength, and postsynaptic cell type,” *J. Neurosci.*, vol. 18, no. 24, pp. 10464–10472, Dec. 1998 [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/9852584
[28] G.-Q. Bi and M.-M. Poo, “Synaptic Modification by Correlated Activity: Hebb’s Postulate Revisited,” *Annual Review of Neuroscience*, vol. 24, no. 1. pp. 139–166, 2001 [Online]. Available: http://dx.doi.org/10.1146/annurev.neuro.24.1.139
[29] C. C. Bell, V. Z. Han, Y. Sugawara, and K. Grant, “Synaptic plasticity in a cerebellum-like structure depends on temporal order,” *Nature*, vol. 387, no. 6630. pp. 278–281, 1997 [Online]. Available: http://dx.doi.org/10.1038/387278a0
[30] W. Gerstner, R. Kempter, J. L. van Hemmen, and H. Wagner, “A neuronal learning rule for sub-millisecond temporal coding,” *Nature*, vol. 383, no. 6595, pp. 76–81, Sep. 1996,

doi: 10.1038/383076a0. [Online]. Available: http://dx.doi.org/10.1038/383076a0
[31] D. O. Hebb, *The organization of behavior: a neuropsychological theory*. John Wiley & Sons, 1949 [Online]. Available: https://books.google.com/books/about/The_organization_of_behavior.html?hl=&id=7NBeyAEACAAJ
[32] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana, "The SpiNNaker Project," *Proceedings of the IEEE*, vol. 102, no. 5. pp. 652–665, 2014 [Online]. Available: http://dx.doi.org/10.1109/jproc.2014.2304638
[33] M. Bouvier *et al.*, "Spiking Neural Networks Hardware Implementations and Challenges," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 15, no. 2. pp. 1–35, 2019 [Online]. Available: http://dx.doi.org/10.1145/3304103
[34] J. Schemmel, D. Briiderle, A. Griibl, M. Hock, K. Meier, and S. Millner, "A wafer-scale neuromorphic hardware system for large-scale neural modeling," *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. 2010 [Online]. Available: http://dx.doi.org/10.1109/iscas.2010.5536970
[35] B. V. Benjamin *et al.*, "Neurogrid: A Mixed-Analog-Digital Multichip System for Large-Scale Neural Simulations," *Proceedings of the IEEE*, vol. 102, no. 5. pp. 699–716, 2014 [Online]. Available: http://dx.doi.org/10.1109/jproc.2014.2313565
[36] P. A. Merolla *et al.*, "Artificial brains. A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, Aug. 2014, doi: 10.1126/science.1254642. [Online]. Available: http://dx.doi.org/10.1126/science.1254642
[37] M. Davies *et al.*, "Loihi: A Neuromorphic Manycore Processor with On-Chip Learning," *IEEE Micro*, vol. 38, no. 1. pp. 82–99, 2018 [Online]. Available: http://dx.doi.org/10.1109/mm.2018.112130359
[38] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Communications of the ACM*, vol. 60, no. 6. pp. 84–90, 2017 [Online]. Available: http://dx.doi.org/10.1145/3065386
[39] D. Ciresan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," *2012 IEEE Conference on Computer Vision and Pattern Recognition*. 2012 [Online]. Available: http://dx.doi.org/10.1109/cvpr.2012.6248110
[40] A. Vaswani *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008 [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf. [Accessed: 15-Oct-2020]
[41] C. Rosset, "Turing-NLG: A 17-billion-parameter language model by Microsoft," 10-Feb-2020. [Online]. Available: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/. [Accessed: 15-Oct-2020]
[42] *STDP-based Spiking Deep Convolutional Neural Networks for Object Recognition*. 2018 [Online]. Available: https://books.google.com/books/about/STDP_based_Spiking_Deep_Convolutional_Ne.html?hl=&id=UXSGtgEACAAJ
[43] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, S. J. Thorpe, and T. Masquelier, "Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks," *Pattern Recognition*, vol. 94. pp. 87–95, 2019 [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2019.05.015
[44] A. Yousefzadeh, T. Serrano-Gotarredona, and B. Linares-Barranco, "Fast Pipeline 128×128 pixel spiking convolution core for event-driven vision processing in FPGAs," *2015 International Conference on Event-based Control, Communication, and Signal Processing (EBCCSP)*. 2015 [Online]. Available: http://dx.doi.org/10.1109/ebccsp.2015.7300698
[45] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "HFirst: A Temporal Approach to Object Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 10, pp. 2028–2040, Oct. 2015, doi: 10.1109/TPAMI.2015.2392947. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2015.2392947

[46] A. Lines *et al.*, “Loihi Asynchronous Neuromorphic Research Chip,” *2018 24th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. 2018 [Online]. Available: http://dx.doi.org/10.1109/async.2018.00018
[47] S. Thorpe and J. Gautrais, “Rank Order Coding,” *Computational Neuroscience*. pp. 113–118, 1998 [Online]. Available: http://dx.doi.org/10.1007/978-1-4615-4831-7_19
[48] S. Thorpe, A. Delorme, and R. Van Rullen, “Spike-based strategies for rapid processing,” *Neural Netw.*, vol. 14, no. 6–7, pp. 715–725, Jul. 2001, doi: 10.1016/s0893-6080(01)00083-1. [Online]. Available: http://dx.doi.org/10.1016/s0893-6080(01)00083-1
[49] M. Mozafari, S. R. Kheradpisheh, T. Masquelier, A. Nowzari-Dalini, and M. Ganjtabesh, “First-Spike-Based Visual Categorization Using Reward-Modulated STDP,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 12. pp. 6178–6190, 2018 [Online]. Available: http://dx.doi.org/10.1109/tnnls.2018.2826721
[50] P. Falez, P. Tirilly, I. M. Bilasco, P. Devienne, and P. Boulet, “Multi-layered Spiking Neural Network with Target Timestamp Threshold Adaptation and STDP,” *2019 International Joint Conference on Neural Networks (IJCNN)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/ijcnn.2019.8852346
[51] J. H. Lee, T. Delbruck, and M. Pfeiffer, “Training Deep Spiking Neural Networks Using Backpropagation,” *Front. Neurosci.*, vol. 10, p. 508, Nov. 2016, doi: 10.3389/fnins.2016.00508. [Online]. Available: http://dx.doi.org/10.3389/fnins.2016.00508
[52] E. Neftci, C. Augustine, S. Paul, and G. Detorakis, “Event-driven random backpropagation: Enabling neuromorphic deep learning machines,” *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2017 [Online]. Available: http://dx.doi.org/10.1109/iscas.2017.8050529
[53] S. Thorpe, D. Fize, and C. Marlot, “Speed of processing in the human visual system,” *Nature*, vol. 381, no. 6582. pp. 520–522, 1996 [Online]. Available: http://dx.doi.org/10.1038/381520a0
[54] S. R. Kheradpisheh and T. Masquelier, “Temporal backpropagation for spiking neural networks with one spike per neuron,” *International Journal of Neural Systems*. 2020 [Online]. Available: http://dx.doi.org/10.1142/s0129065720500276
[55] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11. pp. 2278–2324, 1998 [Online]. Available: http://dx.doi.org/10.1109/5.726791
[56] M. Mozafari, M. Ganjtabesh, A. Nowzari-Dalini, and T. Masquelier, “SpykeTorch: Efficient Simulation of Convolutional Spiking Neural Networks With at Most One Spike per Neuron,” *Frontiers in Neuroscience*, vol. 13. 2019 [Online]. Available: http://dx.doi.org/10.3389/fnins.2019.00625
[57] S. J. Thorpe, R. Guyonneau, N. Guilbaud, J.-M. Allegraud, and R. VanRullen, “SpikeNet: real-time visual processing with one spike per neuron,” *Neurocomputing*, vol. 58–60. pp. 857–864, 2004 [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2004.01.138
[58] W. Maass, “Networks of spiking neurons: The third generation of neural network models,” *Neural Networks*, vol. 10, no. 9. pp. 1659–1671, 1997 [Online]. Available: http://dx.doi.org/10.1016/s0893-6080(97)00011-7
[59] P. U. Diehl, D. Neil, J. Binas, M. Cook, S.-C. Liu, and M. Pfeiffer, “Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing,” *2015 International Joint Conference on Neural Networks (IJCNN)*. 2015 [Online]. Available: http://dx.doi.org/10.1109/ijcnn.2015.7280696
[60] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, “Going Deeper in Spiking Neural Networks: VGG and Residual Architectures,” *Front. Neurosci.*, vol. 13, p. 95, Mar. 2019, doi: 10.3389/fnins.2019.00095. [Online]. Available: http://dx.doi.org/10.3389/fnins.2019.00095
[61] S. Yin *et al.*, “Algorithm and hardware design of discrete-time spiking neural networks based on back propagation with binary activations,” *2017 IEEE Biomedical Circuits and*

Systems Conference (BioCAS). 2017 [Online]. Available: http://dx.doi.org/10.1109/biocas.2017.8325230
[62] A. Tavanaei, Z. Kirby, and A. S. Maida, "Training Spiking ConvNets by STDP and Gradient Descent," *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018 [Online]. Available: http://dx.doi.org/10.1109/ijcnn.2018.8489104
[63] F. Zenke and S. Ganguli, "SuperSpike: Supervised Learning in Multilayer Spiking Neural Networks," *Neural Comput.*, vol. 30, no. 6, pp. 1514–1541, Jun. 2018, doi: 10.1162/neco_a_01086. [Online]. Available: http://dx.doi.org/10.1162/neco_a_01086
[64] A. Tavanaei and A. Maida, "BP-STDP: Approximating backpropagation using spike timing dependent plasticity," *Neurocomputing*, vol. 330. pp. 39–47, 2019 [Online]. Available: http://dx.doi.org/10.1016/j.neucom.2018.11.014
[65] C. Mead, "Neuromorphic electronic systems," *Proceedings of the IEEE*, vol. 78, no. 10. pp. 1629–1636, 1990 [Online]. Available: http://dx.doi.org/10.1109/5.58356
[66] L. Deng, P. Jiao, J. Pei, Z. Wu, and G. Li, "GXNOR-Net: Training deep neural networks with ternary weights and activations without full-precision memory under a unified discretization framework," *Neural Networks*, vol. 100. pp. 49–58, 2018 [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2018.01.010
[67] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *Computer Vision – ECCV 2016*. pp. 525–542, 2016 [Online]. Available: http://dx.doi.org/10.1007/978-3-319-46493-0_32
[68] J. F. Bauters *et al.*, "Planar waveguides with less than 0.1 dB/m propagation loss fabricated with wafer bonding," *Opt. Express*, vol. 19, no. 24, pp. 24090–24101, Nov. 2011, doi: 10.1364/OE.19.024090. [Online]. Available: http://dx.doi.org/10.1364/OE.19.024090
[69] M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," *Phys. Rev. Lett.*, vol. 73, no. 1, pp. 58–61, Jul. 1994, doi: 10.1103/PhysRevLett.73.58. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.73.58
[70] D. Moss, X. Xu, M. Tan, J. Wu, and R. Morandotti, "Photonic perceptron based on a Kerr microcomb for high-speed, scalable, optical neural networks." [Online]. Available: http://dx.doi.org/10.36227/techrxiv.11925225.v1
[71] Y. Tan, S. Chen, and D. Dai, "Polarization-selective microring resonators," *Opt. Express*, vol. 25, no. 4, pp. 4106–4119, Feb. 2017, doi: 10.1364/OE.25.004106. [Online]. Available: http://dx.doi.org/10.1364/OE.25.004106
[72] W. Maass, T. Natschläger, and H. Markram, "Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations," *Neural Computation*, vol. 14, no. 11. pp. 2531–2560, 2002 [Online]. Available: http://dx.doi.org/10.1162/089976602760407955
[73] H. Jaeger, *The "echo State": Approach to Analysing and Training Recurrent Neural Networks*. 2001 [Online]. Available: https://books.google.com/books/about/The_echo_State.html?hl=&id=-3NBHQAACAAJ
[74] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural Netw.*, vol. 20, no. 3, pp. 391–403, Apr. 2007, doi: 10.1016/j.neunet.2007.04.003. [Online]. Available: http://dx.doi.org/10.1016/j.neunet.2007.04.003
[75] D. Mengu, Y. Zhao, N. T. Yardimci, Y. Rivenson, M. Jarrahi, and A. Ozcan, "Misalignment resilient diffractive optical networks," *Nanophotonics*, vol. 9, no. 13. pp. 4207–4219, 2020 [Online]. Available: http://dx.doi.org/10.1515/nanoph-2020-0291
[76] T. Zhou *et al.*, "In situ optical backpropagation training of diffractive optical neural networks," *Photonics Research*, vol. 8, no. 6. p. 940, 2020 [Online]. Available: http://dx.doi.org/10.1364/prj.389553
[77] Y. Chen and J. Zhu, "An optical diffractive deep neural network with multiple frequency-channels," 23-Dec-2019 [Online]. Available: http://arxiv.org/abs/1912.10730. [Accessed: 18-Oct-2020]

[78] X. Lin *et al.*, “All-optical machine learning using diffractive deep neural networks,” *Science*, vol. 361, no. 6406, pp. 1004–1008, Sep. 2018, doi: 10.1126/science.aat8084. [Online]. Available: http://dx.doi.org/10.1126/science.aat8084
[79] Y. Chen, “Express Wavenet -- a low parameter optical neural network with random shift wavelet pattern,” *arXiv.org*. [Online]. Available: https://arxiv.org/abs/2001.01458. [Accessed: 18-Oct-2020]
[80] T. Yan *et al.*, “Fourier-space Diffractive Deep Neural Network,” *Phys. Rev. Lett.*, vol. 123, no. 2, p. 023901, Jul. 2019, doi: 10.1103/PhysRevLett.123.023901. [Online]. Available: http://dx.doi.org/10.1103/PhysRevLett.123.023901
[81] J. Moughames *et al.*, “Three-dimensional waveguide interconnects for scalable integration of photonic neural networks,” *Optica, OPTICA*, vol. 7, no. 6, pp. 640–646, Jun. 2020, doi: 10.1364/OPTICA.388205. [Online]. Available: https://www.osapublishing.org/abstract.cfm?uri=optica-7-6-640. [Accessed: 18-Oct-2020]
[82] M. Miscuglio *et al.*, “All-optical nonlinear activation function for photonic neural networks [Invited],” *Optical Materials Express*, vol. 8, no. 12. p. 3851, 2018 [Online]. Available: http://dx.doi.org/10.1364/ome.8.003851
[83] Y. Shen, N. C. Harris, S. Skirlo, D. Englund, and M. Soljacic, “Deep learning with coherent nanophotonic circuits,” *2017 IEEE Photonics Society Summer Topical Meeting Series (SUM)*. 2017 [Online]. Available: http://dx.doi.org/10.1109/phosst.2017.8012714
[84] I. A. D. Williamson, T. W. Hughes, M. Minkov, B. Bartlett, S. Pai, and S. Fan, “Reprogrammable Electro-Optic Nonlinear Activation Functions for Optical Neural Networks,” *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1. pp. 1–12, 2020 [Online]. Available: http://dx.doi.org/10.1109/jstqe.2019.2930455
[85] T. Ishihara, J. Shiomi, N. Hattori, Y. Masuda, A. Shinya, and M. Notomi, “An Optical Neural Network Architecture based on Highly Parallelized WDM-Multiplier-Accumulator,” *2019 IEEE/ACM Workshop on Photonics-Optics Technology Oriented Networking, Information and Computing Systems (PHOTONICS)*. 2019 [Online]. Available: http://dx.doi.org/10.1109/photonics49561.2019.00008
[86] R. Toole *et al.*, “Photonic Implementation of Spike-Timing-Dependent Plasticity and Learning Algorithms of Biological Neural Systems,” *Journal of Lightwave Technology*, vol. 34, no. 2. pp. 470–476, 2016 [Online]. Available: http://dx.doi.org/10.1109/jlt.2015.2475275
[87] R. Toole and M. P. Fok, “Photonic Implementation of a Neuronal Learning Algorithm based on Spike Timing Dependent Plasticity,” *Optical Fiber Communication Conference*. 2015 [Online]. Available: http://dx.doi.org/10.1364/ofc.2015.w1k.6
[88] M. P. Fok, Y. Tian, D. Rosenbluth, and P. R. Prucnal, “Pulse lead/lag timing detection for adaptive feedback and control based on optical spike-timing-dependent plasticity,” *Opt. Lett.*, vol. 38, no. 4, pp. 419–421, Feb. 2013, doi: 10.1364/OL.38.000419. [Online]. Available: http://dx.doi.org/10.1364/OL.38.000419
[89] S. Xiang *et al.*, “Numerical Implementation of Wavelength-Dependent Photonic Spike Timing Dependent Plasticity Based on VCSOA,” *IEEE Journal of Quantum Electronics*, vol. 54, no. 6. pp. 1–7, 2018 [Online]. Available: http://dx.doi.org/10.1109/jqe.2018.2879484
[90] C. Mesaritakis, M. Skontranis, G. Sarantoglou, and A. Bogris, “Micro-Ring-Resonator Based Passive Photonic Spike-Time-Dependent-Plasticity Scheme for Unsupervised Learning in Optical Neural Networks,” *Optical Fiber Communication Conference (OFC) 2020*. 2020 [Online]. Available: http://dx.doi.org/10.1364/ofc.2020.t4c.2
[91] M. Y.-S. Fang, S. Manipatruni, C. Wierzynski, A. Khosrowshahi, and M. R. DeWeese, “Design of optical neural networks with component imprecisions,” *Opt. Express*, vol. 27, no. 10, pp. 14009–14029, May 2019, doi: 10.1364/OE.27.014009. [Online]. Available: http://dx.doi.org/10.1364/OE.27.014009
[92] A. N. Tait *et al.*, “Silicon Photonic Modulator Neuron,” *Physical Review Applied*, vol. 11, no. 6. 2019 [Online]. Available: http://dx.doi.org/10.1103/physrevapplied.11.064043
[93] Q. Cheng, J. Kwon, M. Glick, M. Bahadori, L. P. Carloni, and K. Bergman, “Silicon

Photonics Codesign for Deep Learning," *Proceedings of the IEEE*. pp. 1–22, 2020 [Online]. Available: http://dx.doi.org/10.1109/jproc.2020.2968184

[94] D. Moss, X. Xu, M. Tan, J. Wu, and R. Morandotti, "A single photonic perceptron based on a soliton crystal microcomb for scalable, high speed, optical neural networks." [Online]. Available: http://dx.doi.org/10.36227/techrxiv.11925225.v3

[95] A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Broadcast and Weight: An Integrated Network For Scalable Photonic Spike Processing," *Journal of Lightwave Technology*, vol. 32, no. 21. pp. 4029–4041, 2014 [Online]. Available: http://dx.doi.org/10.1109/jlt.2014.2345652

[96] A. N. Tait *et al.*, "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific Reports*, vol. 7, no. 1. 2017 [Online]. Available: http://dx.doi.org/10.1038/s41598-017-07754-z

[97] P. Y. Ma, A. N. Tait, T. F. de Lima, C. Huang, B. J. Shastri, and P. R. Prucnal, "Photonic independent component analysis using an on-chip microring weight bank," *Optics Express*, vol. 28, no. 2. p. 1827, 2020 [Online]. Available: http://dx.doi.org/10.1364/oe.383603

[98] A. N. Tait *et al.*, "Feedback control for microring weight banks," *Optics Express*, vol. 26, no. 20. p. 26422, 2018 [Online]. Available: http://dx.doi.org/10.1364/oe.26.026422

[99] J. Xiang, A. Torchy, X. Guo, and Y. Su, "All-Optical Spiking Neuron Based on Passive Microresonator," *Journal of Lightwave Technology*. pp. 1–1, 2020 [Online]. Available: http://dx.doi.org/10.1109/jlt.2020.2986233

[100] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, "Large-Scale Optical Neural Networks Based on Photoelectric Multiplication," *Physical Review X*, vol. 9, no. 2. 2019 [Online]. Available: http://dx.doi.org/10.1103/physrevx.9.021032

[101] T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," *Optica*, vol. 5, no. 7. p. 864, 2018 [Online]. Available: http://dx.doi.org/10.1364/optica.5.000864

[102] N. Passalis, G. Mourgias-Alexandris, A. Tsakyridis, N. Pleros, and A. Tefas, "Training Deep Photonic Convolutional Neural Networks With Sinusoidal Activations," *IEEE Transactions on Emerging Topics in Computational Intelligence*. pp. 1–10, 2020 [Online]. Available: http://dx.doi.org/10.1109/tetci.2019.2923001

[103] V. Bangari *et al.*, "Digital Electronics and Analog Photonics for Convolutional Neural Networks (DEAP-CNNs)," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1. pp. 1–13, 2020 [Online]. Available: http://dx.doi.org/10.1109/jstqe.2019.2945540

[104] A. Mehrabian, M. Miscuglio, Y. Alkabani, V. J. Sorger, and T. El-Ghazawi, "A Winograd-Based Integrated Photonics Accelerator for Convolutional Neural Networks," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 26, no. 1. pp. 1–12, 2020 [Online]. Available: http://dx.doi.org/10.1109/jstqe.2019.2957443

[105] H. Bagherian, S. Skirlo, Y. Shen, H. Meng, V. Ceperic, and M. Soljacic, "On-Chip Optical Convolutional Neural Networks," 09-Aug-2018 [Online]. Available: http://arxiv.org/abs/1808.03303. [Accessed: 18-Oct-2020]

[106] A. Mehrabian, Y. Al-Kabani, V. J. Sorger, and T. El-Ghazawi, "PCNNA: A Photonic Convolutional Neural Network Accelerator," *2018 31st IEEE International System-on-Chip Conference (SOCC)*. 2018 [Online]. Available: http://dx.doi.org/10.1109/socc.2018.8618542

[107] J. Feldmann, "Parallel convolution processing using an integrated photonic tensor core," *arXiv*. [Online]. Available: https://arxiv.org/abs/2002.00281. [Accessed: 18-Oct-2020]

[108] A. Driessen, "Microresonators As Building Blocks For VLSI Photonics," *AIP Conference Proceedings*. 2004 [Online]. Available: http://dx.doi.org/10.1063/1.1764011

[109] V. Sitzmann, J. N. P. Martel, A. W. Bergman, D. B. Lindell, and G. Wetzstein, "Implicit Neural Representations with Periodic Activation Functions," 17-Jun-2020 [Online]. Available: http://arxiv.org/abs/2006.09661. [Accessed: 28-Oct-2020]

6. Appendices

Appendix A: Analysis of temporally-encoded WDM signals and second order-dispersion coding

For each time step that the photodetector can stably detect a signal, only one piece of temporal data can be encoded. We need to produce a minimum of *n* time domain values, *n* being the number of different wavelengths that have been scaled. Then, these *n* different values need to be skewed such that the diagonal matrix elements align into one time step, resulting in a total time per calculation of 2*n* - 1 time steps.

$$\mathbf{X}\times\mathbf{W}^T=\begin{pmatrix} w(1)\cdot x(1) & w(1)\cdot x(2) & w(1)\cdot x(3) & \cdots & w(1)\cdot x(49) \\ w(2)\cdot x(1) & w(2)\cdot x(2) & w(2)\cdot x(3) & \cdots & w(2)\cdot x(49) \\ w(3)\cdot x(1) & w(3)\cdot x(2) & w(3)\cdot x(3) & \cdots & w(3)\cdot x(49) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w(49)\cdot x(1) & w(49)\cdot x(2) & w(49)\cdot x(3) & \cdots & w(49)\cdot x(49) \end{pmatrix}$$

$$\Big\downarrow \text{Second-order time dispersion}$$

$$\left(\begin{array}{ccccccccc} & & & & w(1)\cdot x(1) & \cdots & w(1)\cdot x(47) & w(1)\cdot x(48) & w(1)\cdot x(49) \\ & & & w(2)\cdot x(1) & w(2)\cdot x(2) & \cdots & w(2)\cdot x(48) & w(2)\cdot x(49) & \\ & & w(3)\cdot x(1) & w(3)\cdot x(2) & w(3)\cdot x(3) & \cdots & w(3)\cdot x(49) & & \\ & \iddots & \vdots & \vdots & \vdots & \iddots & & & \\ w(49)\cdot x(1) & \cdots & w(49)\cdot x(47) & w(49)\cdot x(48) & w(49)\cdot x(49) & & & & \end{array}\right)$$

Figure A1: Effects of second order dispersive effects on temporal signals. Here x is the time domain vector, whilst w is the weight vector, with each scalar in w applied to a different wavelength. Adjusted from [70]

Increasing the number of wavelengths onto which data can be encoded will increase the number of parallel calculations performed by the photodetector, but also increase the amount of time needed to perform each computation.
Moss et al. used 49 different wavelengths, and a photodetector with 87ps time steps, meaning that each calculation will take (49*2 - 1) * 87 picoseconds = 8.44 nanoseconds.

Doubling the amount of wavelengths used to 98, then you would need (98*2 - 1) * 87 picoseconds = 17ns to perform a calculation.

Assuming the resolution of the scaling of each independent wavelength is 8 bits, as per Moss et al., then the first situation performs at 49*2^8/(8.44 nanoseconds) = 1.49 Tb/s, and the second situation performs at (98*2^8)/(17ns) = 1.47 Tb/s. This demonstrates a lack of scalability with extra wavelengths, and so the proposed architecture does not utilise the additional wavelengths provided by WDM effectively. Indeed, extending this to its logical minimum, with only 2 wavelengths and a 2 input temporal signal, we have (2*2 - 1) *87 picoseconds for the calculation, or 261 picoseconds, and 2*2^8/261 picoseconds = 1.96 Tb/s.
Even if we assume a 200 picosecond additional delay for the second order dispersive effect(as proposed by Moss et al.), the results are not much better.
49 wavelengths takes 8.64 nanoseconds, and thus has 1.45 Tb/s, 98 wavelengths takes 17.165 nanoseconds, and has 1.46 Tb/s, and 2 wavelengths has 1.11 Tb/s. For increasing the number of wavelengths (theoretically) from 2 to 98, we only increase the bandwidth of this network by a factor of 1.3.

The bandwidth of this perceptron can be defined as: $Bandwidth = \dfrac{n \cdot r}{t_p \cdot ((2 \cdot n) - 1) + t_d}$

where *r* is the resolution of each calculation(ie 8 bits of resolution = 256), *n* is the number of independent wavelengths, t_p is the sampling time of the photodetector and t_d is the delay time of the second order dispersive effect.

Assuming $t_d \rightarrow 0$, n>>0, we have $Bandwidth \simeq \dfrac{r}{2 \cdot t_p}$
and so the bandwidth of the system is not increased substantially by increases in the number of wavelengths, as for each additional wavelength there is a 2*t_p *additional delay* as a result of the photodetector only detecting one timed slot for any n wavelengths.
Thus, it seems unlikely that utilising the time-domain to encode signals in this method will be beneficial in creating a high-bandwidth ONN, as it scales poorly with the benefits of WDM signals.

Appendix B: Overview diagram of an example weight update circuit which stores the weight of a neuron as a voltage on a capacitor, and updates it via stochastic gradient descent.

The blue area performs the balanced photodetection and summation of loss derivative signals, and each neuron only needs one instance of these components. Then, each WDM channel that the neuron can operate upon requires one instance of the components contained within the larger gray box. The pink box is the wavelength specific incoming intensity accumulation circuit, the white box is a RF multiplier which computes the second term in the SGD weight update rule, namely multiplying the loss derivative value by the intensity value accumulated, and then multiplying this by an externally set ‘learning rate’ parameter, equal to the size of the steps taken in a direction when performing gradient descent. The yellow box is a simple unity summer, which adds the stored value and this RF multiplied value together, and the result of this computation is stored in the dark green box.

Upon an external pulse, this dark green box is switched and disconnected from the summing op amp, and the light green box which corresponds to the stored weight values is then updated to store the value held by the temporary storage capacitor in the light green box. When the external pulse

Upon an external update signal, the dark green box is disconnected from the summing op amp, and the light green box is switched to connect to the dark green box, which shuttles the value stored at the output of the summing op amp to the sample and hold circuit in the light green box. This is equivalent to moving the new weight from a temporary storage capacitor to the regular circuit for storing weights. Once the pulse returns to logical zero, the dark green circuit is reconnected to the summing op amp, and the light green sample and hold is disconnected from the temporary storage capacitor. As a result, the circuit begins calculating the next SGD update value immediately.

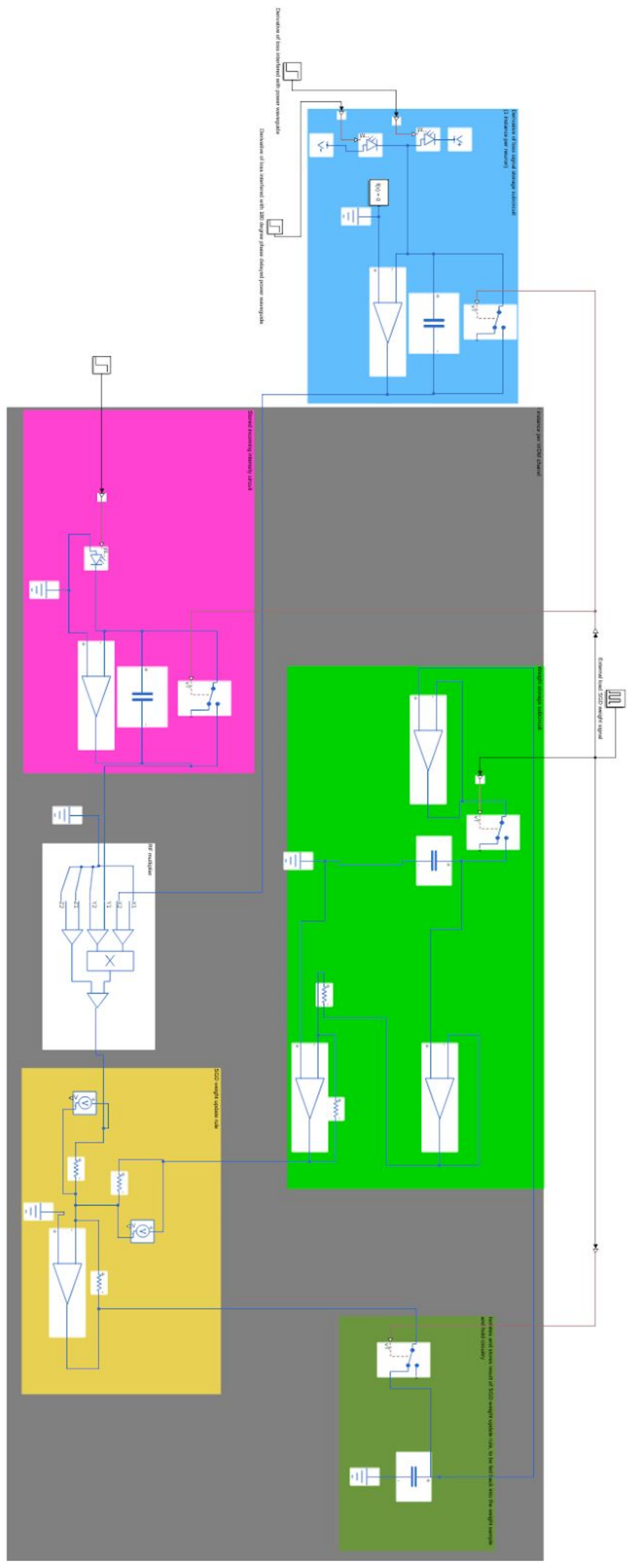

Derivative of loss interfered with power waveguide
Derivative of loss interfered with 180 degree phase delayed power waveguide
Derivative of loss signal storage subcircuit (1 instance per neuron)
Stored incoming intensity circuit
RF multiplier
SGD weight update rule
External load SGD weight signal

Figure B1: The entire weight update circuit with inbuilt stochastic gradient descent

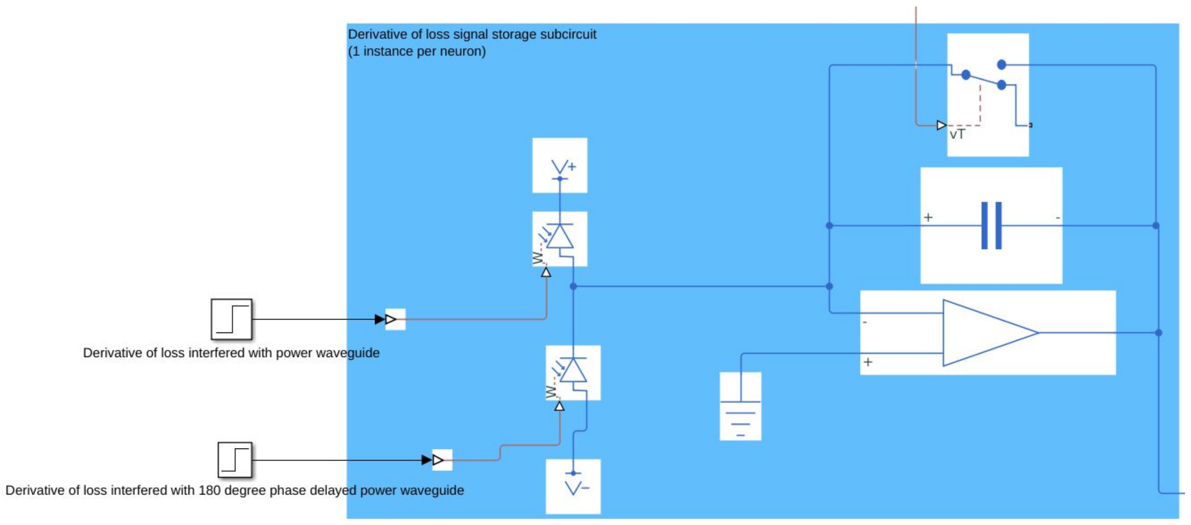

Derivative of loss signal storage subcircuit
(1 instance per neuron)
V+
V-
vT
W
Derivative of loss interfered with power waveguide
Derivative of loss interfered with 180 degree phase delayed power waveguide

Figure B2: Balanced photodetection circuit to store the sign and magnitude of the derivative of the loss signal with respect to this neuron's outputs

The circuit shown in Figure B2 entails the proposed balanced photodetection accumulation circuit, which shall be used to store the sum of each example's derivative of the global loss function of the network with respect to this neuron's output. The single-pole double-throw switch above the capacitor is switched by the same external signal which triggers the application of the stochastic gradient descent weight update rule, and the switch then resets this accumulation capacitor.

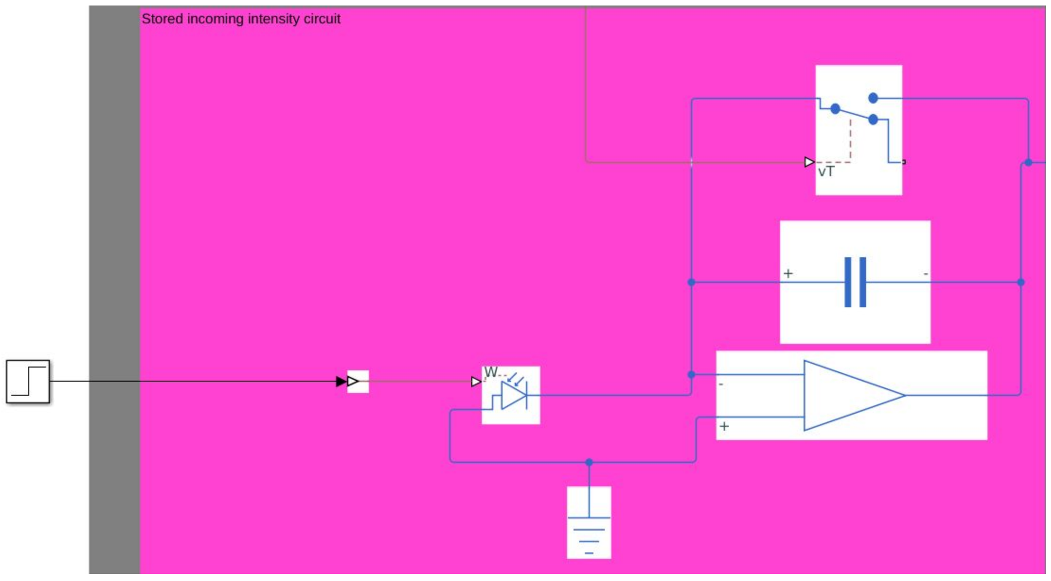


Figure B3: Accumulator to store the sum of incoming light intensity for a single WDM channel. The switch which drains the capacitor is used to reset the accumulated value after each SGD weight update is applied.

Since the input to each neuron is the output from a prior neuron, and all neurons in the

proposed photo-electric neural network have a ReLU activation function, there is no need for balanced photodetection, as all inputs will be 0 or greater symbolically.
A single-pole, double-throw switch is used to clear the stored accumulated value whenever the SGD weight update rule is applied, as it is triggered by the same external signal.

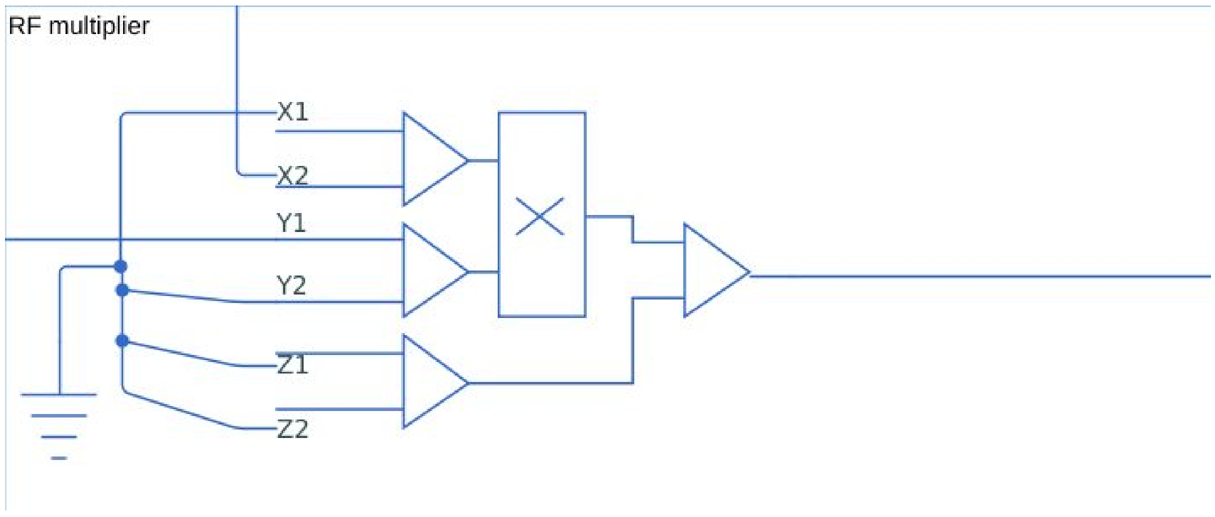


Figure B4: RF multiplier to multiply the local derivative signal by the input to the neuron on a specific WDM channel.

The subcircuit shown in Figure B4 then calculates the product of the local derivative signal and the local incoming intensity signal, and scales this value by the learning rate desired (this is the right hand term in Equation 2).

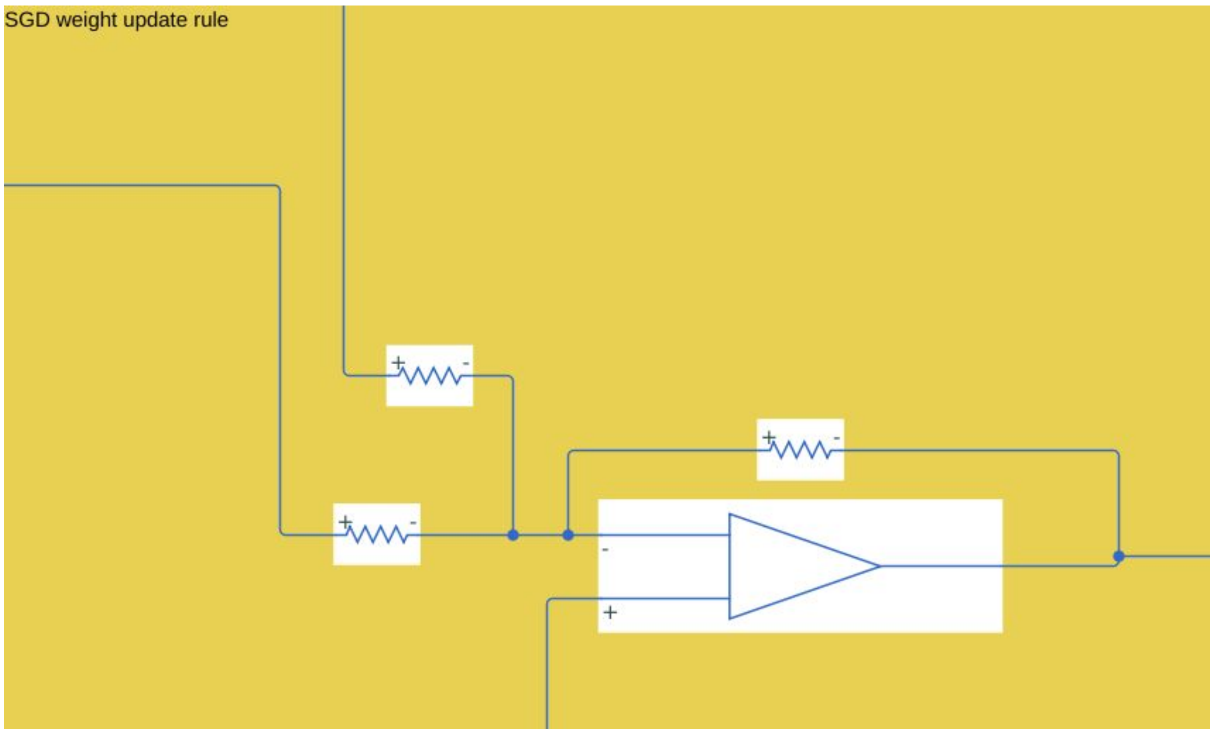
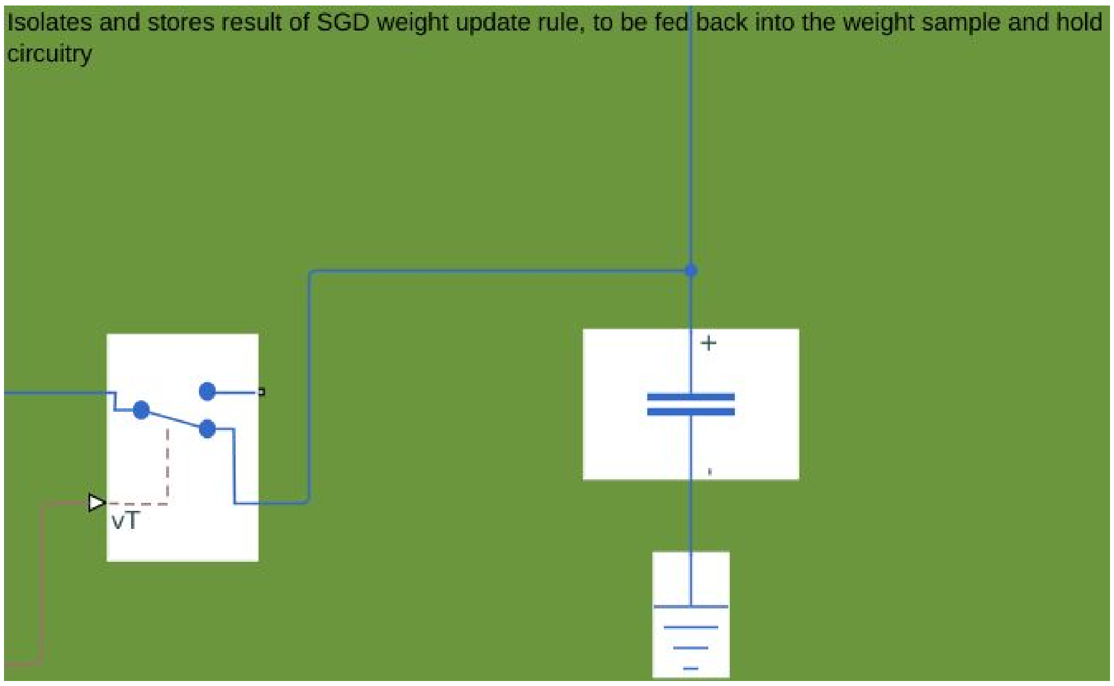


Figure B5: Active summer to calculate the sum of the current stored weight and the value it should change by.



Isolates and stores result of SGD weight update rule, to be fed back into the weight sample and hold circuitry
vT
+

Figure B6: Upon the presence of the external signal to store the current weight value, the output of the subcircuit shown in B5 is loaded onto the capacitor shown in Figure B6.

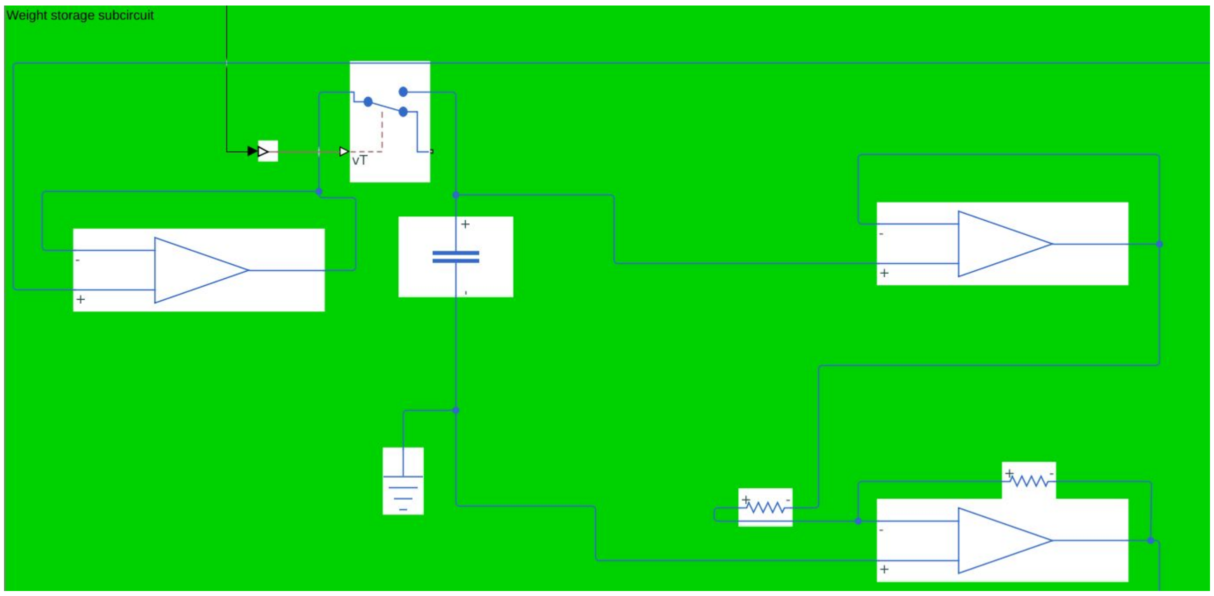


Figure B7: The value stored upon the capacitor shown in Figure B6 is sampled by the sample and hold circuit above upon the presence of an external signal. When this signal is removed, the new calculated weight value is held by the circuit above, and fed back into the op-amp summer shown in Figure B5, such that the next SGD weight update step can be calculated as necessary.

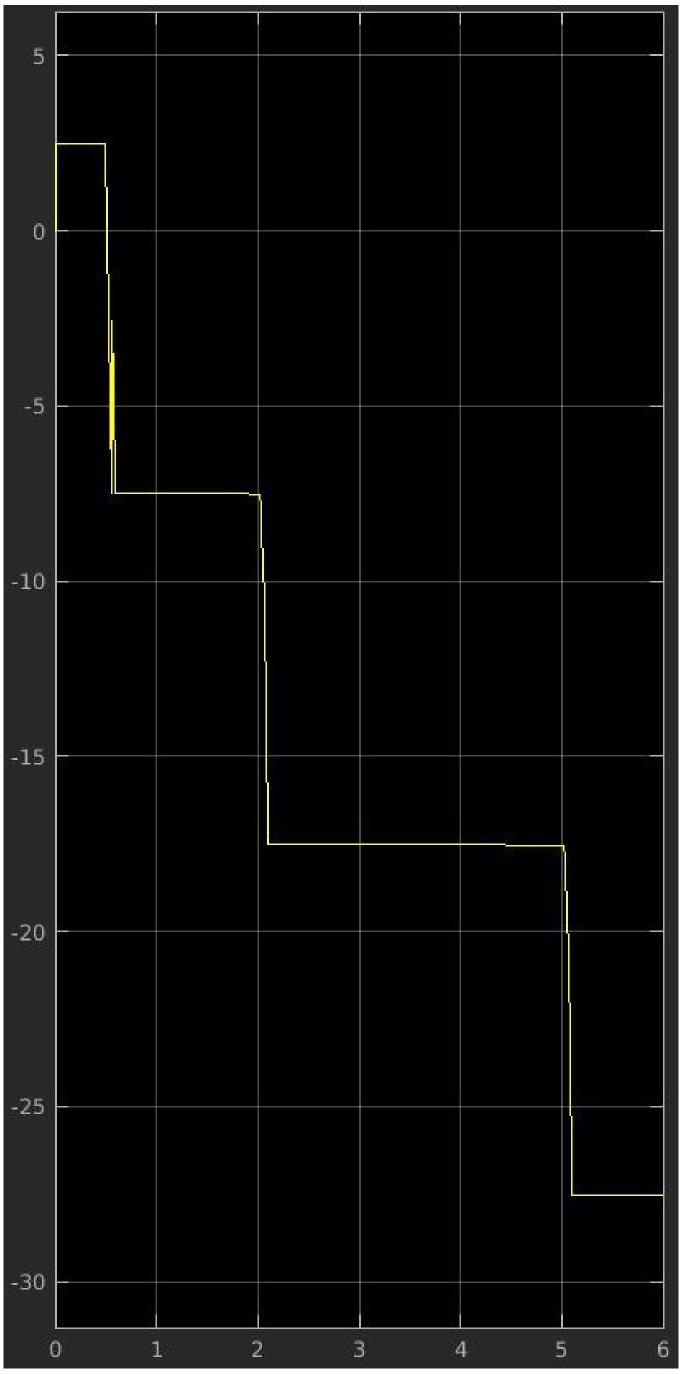
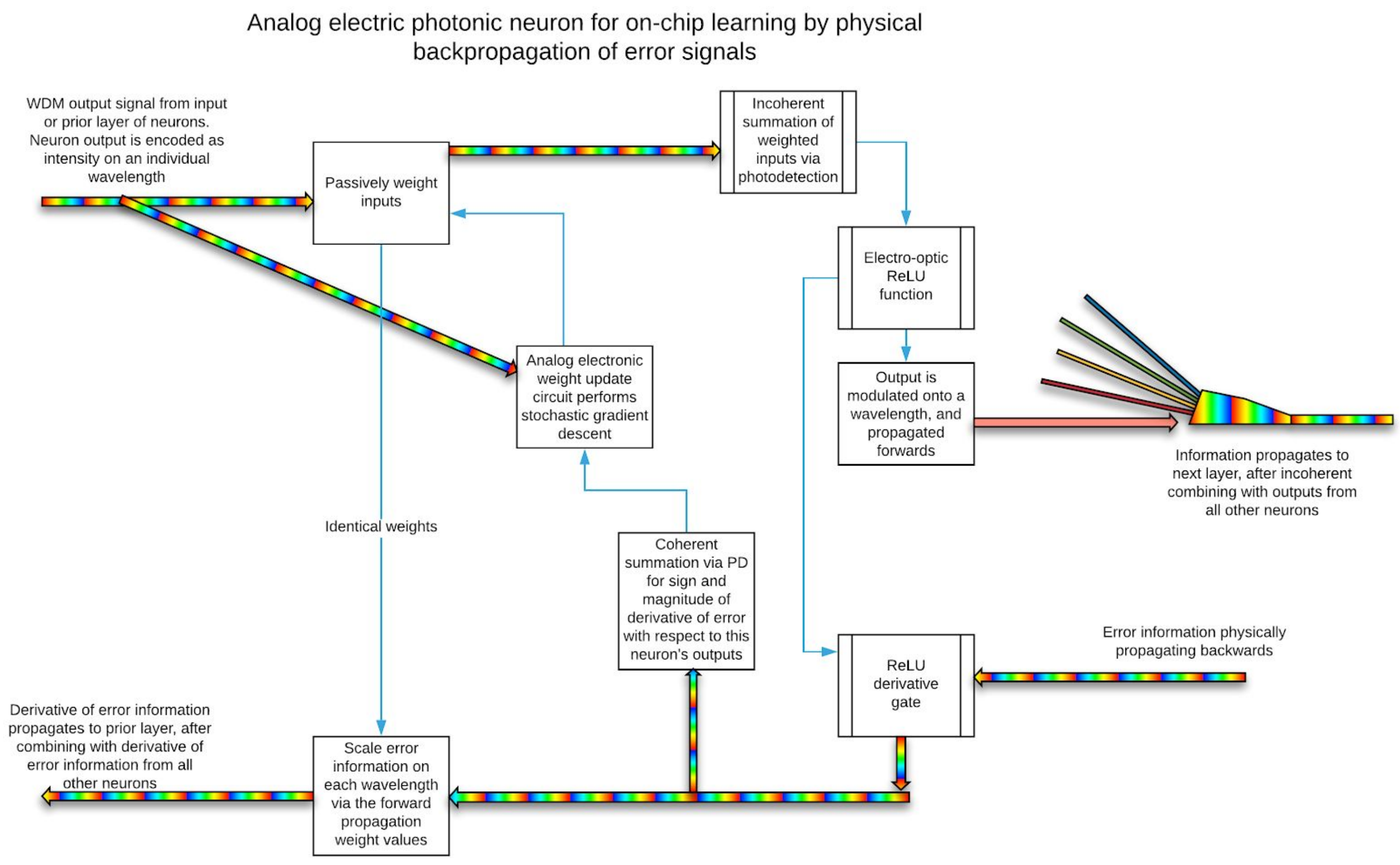
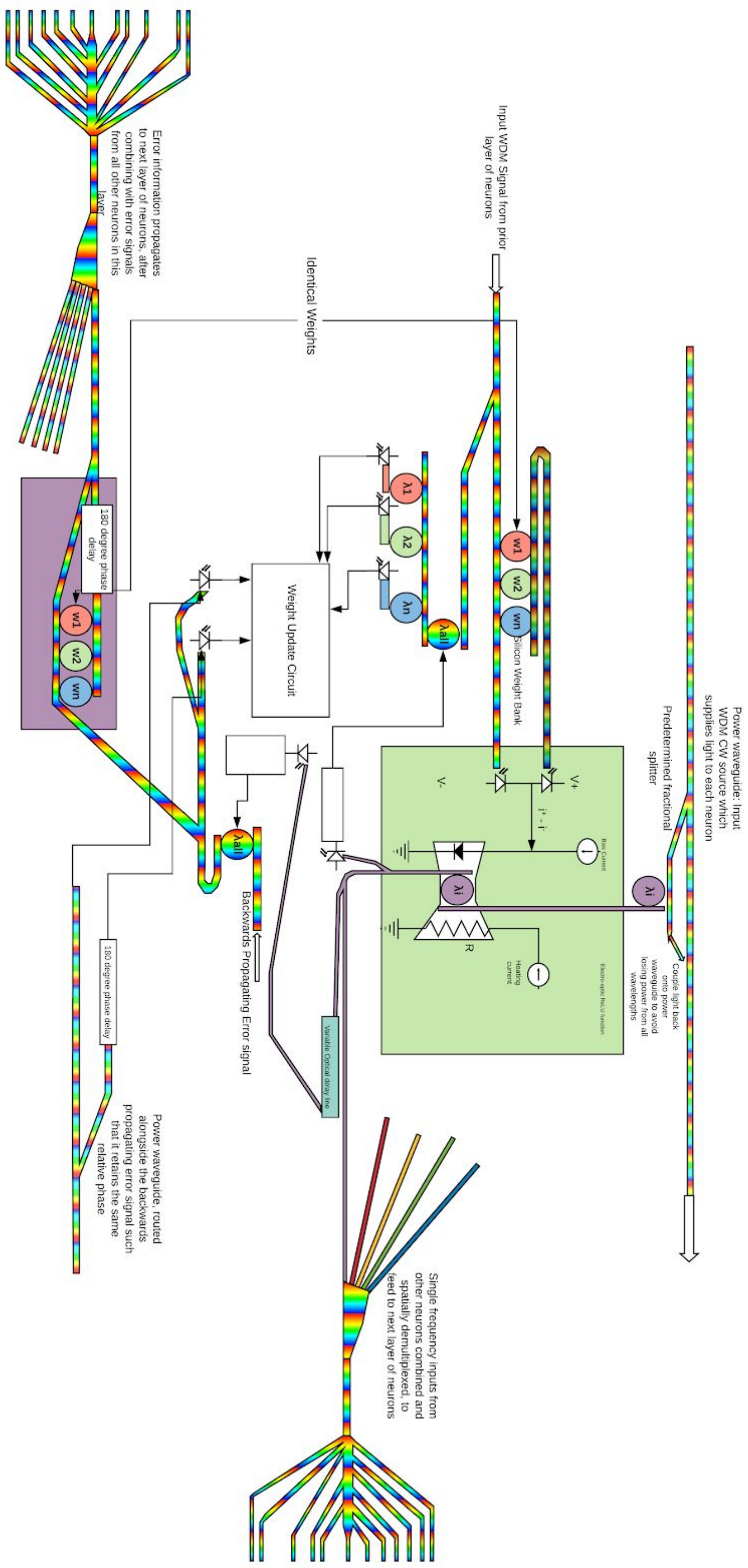


Figure B8: Sample outputs from a simulation of the weight update circuit, for a starting weight of -2.5, a learning rate of 10, an accumulated loss value of 1, with the SGD update rule applied thrice. The mathematical result should be 27.5 (axis inverted due to nominal capacitor storage sign)

Appendix C1: Photonic ReLU neuron abstract diagram


Analog electric photonic neuron for on-chip learning by physical backpropagation of error signals
WDM output signal from input or prior layer of neurons. Neuron output is encoded as intensity on an individual wavelength
Passively weight inputs
Incoherent summation of weighted inputs via photodetection
Electro-optic ReLU function
Output is modulated onto a wavelength, and propagated forwards
Information propagates to next layer, after incoherent combining with outputs from all other neurons
Analog electronic weight update circuit performs stochastic gradient descent
Identical weights
Coherent summation via PD for sign and magnitude of derivative of error with respect to this neuron's outputs
Error information physically propagating backwards
ReLU derivative gate
Derivative of error information propagates to prior layer, after combining with derivative of error information from all other neurons
Scale error information on each wavelength via the forward propagation weight values

Appendix C2: Photonic ReLU Neuron detailed diagram



Error information propagates to next layer of neurons, after combining with error signals from all other neurons in this layer
Input WDM Signal from prior layer of neurons
Identical Weights
180 degree phase delay
w1
w2
wn
λ1
λ2
λn
λall
Weight Update Circuit
Silicon Weight Bank
Power waveguide: Input WDM CW source which supplies light to each neuron
Predetermined fractional splitter
V+
V-
i⁺ - i⁻
Bias Current
λi
R
Heating current
Electro-optic ReLU function
Couple light back onto power waveguide to avoid losing power from all wavelengths
Backwards Propagating Error signal
180 degree phase delay
Variable Optical delay line
Power waveguide, routed alongside the backwards propagating error signal such that it retains the same relative phase
Single frequency inputs from other neurons combined and spatially demultiplexed, to feed to next layer of neurons