ENPH 455 Final Report

Ultrafast Digital Electronics and Analog Photonics (DEAP) for Convolutional Neural Networks and Image Processing

Viraj Bangari

Supervisor: Dr. Bhavin J. Shastri

2019-03-27

Department of Physics, Engineering Physics & Astronomy Queen's University

Abstract

Convolutions are computationally intensive mathematical operations used in convolutional neural networks (CNNs) and image processing. Convolution operations are typically delegated to GPUs due to their ability to highly parallelize matrix multiplication operations. In recent years, silicon photonics has shown promise in being the next generation of computing hardware that can operate at ultrafast speeds. In particular, the neuromorphic photonic broadcast-and-weight architecture has been able to implement recurrent neural networks while operating at gigahertz frequencies. Inspired by the principles of broadcast-and-weight, this thesis proposes two photonic architectures that are capable of performing convolution operations. The first architecture, called DEAP, is specialized for implementing convolutional neural networks whereas the second, DEAP-GIP, is specialized for general-purpose image processing tasks. DEAP is estimated to perform convolutions between 2.8 and 14 times faster than a GPU while roughly using 0.75 times the energy consumption. Additionally, DEAP-GIP is estimated to operate 4.6 to 68 times faster than a GPU while using 2.3 times the energy consumption. The largest bottlenecks for both of these architectures are from the I/O interfacing with digital systems via digital-to-analog converters (DACs) and analog-to-digital converters (ADCs). If photonic DACs [1] and ADCs [2] are to be built with higher bit-precisions, the speedup over GPUs could be even higher. Overall, silicon photonics has the potential to outperform conventional electronic hardware for convolutions while having the ability to scale up in the future.

Contents

Abstract i

List of Tables iv

List of Figures iv

Glossary of Terms v

Acknowledgements vi

1 Introduction 1

2 Convolution Background 2

2.1 Convolutions for Image Processing 2

2.2 Convolutional Neural Networks 3

2.2.1 The Convolutional Layer using Matrix Multiplications 4

3 Silicon Photonics Background 5

3.1 Waveguides 5

3.2 Microring Resonators 6

3.2.1 Wavelength Division Multiplexing and Optical Modulation 8

4 Dot Products using Photonics 9

4.1 Overall Architecture 9

4.2 Representing Negative Inputs 11

4.3 Throughput and Size Estimation 12

5 Photonic Convolutions for CNNs 13

5.1 A Top-down View 13

5.2 Producing a Single Convolved Pixel 14

5.3 Performing a Full Convolution 18

5.4 Throughput and Energy Estimation 20

6 Photonic Convolutions for General Image Processing 21

6.1 Rationale for a second approach 21

6.2 Input Representation 22

6.3 Kernel Representation 25

6.4 Performing a Convolution 27

6.5 Throughput and Energy Estimation 29

7 Benchmarks 30

7.1 Proof of Concept Simulation 30

7.2 Estimated DEAP Performance 31

7.3 Estimated DEAP-GIP Performance 33

8 Conclusion 34
9 References 36
Appendix A.1 Benchmarking Calculations for DEAP and DEAP-GIP 39
Appendix A.2 Selected DEAP simulator code 39
Appendix B. A Note on Semiconductor Optical Amplifiers 40
Appendix C. Statement of Work 41
Appendix D. Floating Point Arithmetic using IEEE 754 41

List of Tables

Table 1 - Summary of Convolutional Parameters 4
Table 2 - DEAP bounding parameters 14
Table 3 - Required components for DEAP input representation 16
Table 4 – DEAP required components for applying kernel 17
Table 5 – DEAP-GIP bounding parameters 21
Table 6 - Required components for DEAP-GIP input representation 24
Table 7 - DEAP-GIP required components for applying kernel 27
Table 8 - Benchmarked GPUs with power consumption 31
Table 9 - Benchmarking parameters for DEAP 39
Table 10 - Benchmarking parameters for DEAP-GIP 39

List of Figures

Figure 1 - Convolutions in image processing 3
Figure 2 - Performing a convolution as a matrix-matrix multiplication. Image was modified from [12]. 5
Figure 3 - Left: Scanning electron microscope image of silicon waveguide. Right: TE propagation of a similar waveguide. Both images taken from [13]. 6
Figure 4 - All-pass and add-drop MRRs 6
Figure 5 - Phase dependent transfer of MRR through port and drop port lines 8
Figure 6 – An electro-optic architecture for performing dot products 11
Figure 7 - Block diagram of DEAP architecture 14
Figure 8 – DEAP input representation with photonics 15
Figure 9 – Applying the kernel with photonics for DEAP 17
Figure 10 – Passive voltage adder design 18
Figure 11 - Performing a convolution using DEAP 19
Figure 12 - Performing a convolution with two convolutional units 20
Figure 13 - Wavelength assignment to pixel locations 23
Figure 14 - DEAP-GIP input representation with photonics 24
Figure 15 - DEAP-GIP input assignment to weight banks 25
Figure 16 – Applying the kernel for DEAP-GIP 27
Figure 17 - Performing a convolution using DEAP-GIP 28
Figure 18 - DEAP and DEAP-GIP Simulation Results 31
Figure 19 - Estimated DEAP convolutional runtime compared to actual GPU runtimes from DeepBench benchmarks 32
Figure 20 - Estimated DEAP-GIP convolutional runtime compared to actual GPU runtimes from DeepBench benchmarks 34

Glossary of Terms

ADC – Analog-to-Digital Converter

CNN – Convolutional Neural Network

DAC – Digital-to-Analog Converter

DEAP – Digital Electronics and Analog Photonics

DEMUX – Demultiplexer

GDDR – Graphics Double Data Rate

GPU – Graphics Processing Unit

MRR – Microring Resonator

MUX – Multiplexer

PD – Photodiode

PWB – Photonic Weight Bank

SDRAM – Synchronous Dynamic Random-Access Memory

TIA – Transimpedance Amplifier

WDM – Wavelength Division Multiplexing

Acknowledgements

I would like to thank my supervisor, Dr. Bhavin Shastri, for his tremendous support throughout the duration of my thesis. Bhavin's immense breadth of knowledge, his relaxed attitude, and genuine care for others make the even most of daunting tasks manageable. I have learned so much from Bhavin in such a short time-frame, and I know that we will continue to stay in touch in future years to come.

Second, I would like to thank Dr. Bicky Marquez and Heidi Miller for their support and dedication towards my thesis over the past year. I cannot thank them enough for taking my ideas seriously and spending the time to provide detailed and valuable feedback to even the most minute of details. I also want to acknowledge Heidi for being the one who came with the acronym "DEAP" used in the title of the thesis.

Finally, I would like to thank my loving partner and best friend, Gloria Li. Thank you for supporting me as I switched from Computing to Engineering Physics. I know that the past four years have sometimes been challenging, so I appreciate you for sticking by me and never giving up. Over the past four years, you have consistently kept me grounded, mature and whole. I cannot thank you enough for all the hours of support you have given me.

1 Introduction

Convolutions are fundamental mathematical operations used in image processing and convolutional neural networks (CNNs). In image processing, convolutions are used for creating visual effects such as a gaussian blur or for more complicated algorithms like Canny edge detection [3]. In CNNs, convolutions are used for state-of-the-art computer vision and machine learning algorithms such as real-time object detection [4, 5]. One of the challenges with convolutions is that they are computationally intensive operations, taking up 86% to 94% of execution time for CNNs [6]. For heavy workloads, convolutions are typically run on graphical processing units (GPUs), as they are able to perform many mathematical operations in parallel.

An emerging alternative to GPU computing is optical computing using silicon photonics. Silicon photonics is a technology that allows for the implementation of photonic circuits by using the existing complementary-metal-oxide-semiconductor (CMOS) platform for electronics [7]. In recent years, the silicon photonic based "broadcast-and-weight" architecture has been shown to perform multiply-accumulate operations at frequencies up to five times faster than conventional electronics [8]. Therefore, there is motivation to explore how photonics can be used to perform convolutions, and how it compares to GPU-based implementations.

The goal of the thesis is to design a photonic architecture capable of performing convolutions at gigahertz operating frequencies. This architecture should be made up of existing and relatively mature photonic and electronic components. The speed and energy consumption of the photonic architecture will need to be estimated and benchmarked against a series of modern, high-end GPUs. Though overall the architecture can be designed to be

general purpose, the speed and energy estimation should use realistic hardware implementation values. Finally, a simulator will be written to verify that the architecture is able to properly perform a convolution.

2 Convolution Background

2.1 Convolutions for Image Processing

A convolution of two discrete domain functions f and g is defined by:

$$(f * g)[t] = \sum_{t=-\infty}^{\infty} f[\tau]g[t-\tau] \tag{1}$$

In digital image processing, the convolution of an image $\boldsymbol{D}$ with a kernel $\boldsymbol{F}$ produces a convolved image $\boldsymbol{O}$. An image is represented as a matrix of numbers with dimensionality $H \times W$ where H and W are the height and width of the image. Each element of a matrix represents the intensity of a pixel at that particular spatial location. A kernel is a matrix of real numbers with dimensionality $R \times R$. The value of a particular convolved pixel is defined by:

$$\boldsymbol{O}[i,j] = \sum_{k=0}^{R-1}\sum_{l=0}^{R-1} \boldsymbol{F}[k,l]\boldsymbol{D}[i+k, j+l] \tag{2}$$

Using matrix slicing notation, equation (2) can be equivalent represented as a dot product of two vectorized matrices:

$$\boldsymbol{O}[i,j] = \text{vec}(\boldsymbol{F})^T \cdot \text{vec}(\boldsymbol{D}[i\!:\! i+R, j\!:\! j+R])^T \tag{3}$$

A convolution operation can apply various effects to an input image based on the values of a kernel, as demonstrated in Figure 1. If the image has multiple colour channels, the same kernel is applied to each channel filter. A convolution reduces the dimensionality of the input

image to $H - R + 1 \times H - R + 1$, so a padding of zero values is normally applied around the edges of the input image to counteract this.

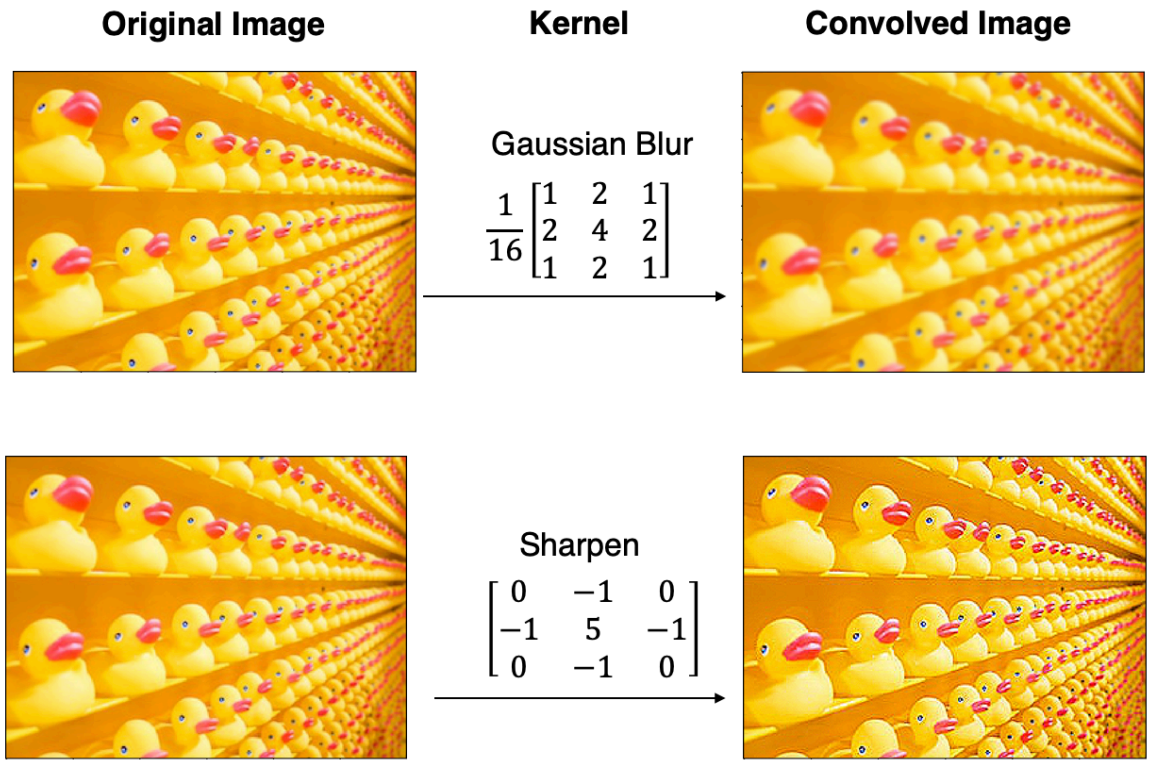

Figure 1 - Convolutions in image processing

2.2 Convolutional Neural Networks

CNNs are a type of neural network that were developed for image recognition tasks. A CNN consists of some combination of convolutional, nonlinear, pooling and fully connected layers [9]. Convolutional neural networks are trained by changing the values of the kernels, analogous to how feedforward neural networks are trained by changing the weighted connections [10]. For CNNs, a convolution operation is defined as:

$$O[i,j] = \text{vec}(\boldsymbol{F})^T \cdot \text{vec}(\boldsymbol{D}[i * S{:}\, i * S + R, j * S{:}\, j * S + R, :])^T \quad (4)$$

where the input D has dimensionality $H \times W \times C$, kernel F has dimensionality $R \times R \times C$ and C refers to the number of channels within the input image. The additional parameter S is

referred to as the "stride" of the convolution. This convolution is similar to the one described in equation (3), except that the outputs from each channel are summed together in the end and that the stride parameter is always equal to 1 in image processing. The dimensionality of the output feature is $\left\lceil \frac{H-R}{S} + 1 \right\rceil \times \left\lceil \frac{W-R}{S} + 1 \right\rceil \times K$, where K is the number of different kernels applied to an image. Table 1 contains a summary of all the convolutional parameters described so far.

Table 1 - Summary of Convolutional Parameters

Parameter	Meaning
N	Number of input images
H	Height of input image including padding
W	Width of input image including padding
C	Number of input channels
R	Edge length of kernel
K	Number of kernels
S	Stride

2.2.1 The Convolutional Layer using Matrix Multiplications

A GPU is a specialized hardware unit that is capable of performing a single mathematical operation on large amounts of data at once. This parallelization allow GPUs to compute matrix-matrix multiplication at speeds much higher than a CPU [11]. The convolution operation defined by (4) can be generalized into a single matrix-matrix multiplication [12]. This is shown in Figure 2, where the K kernels are transformed into a matrix of dimensionality $K \times CR^2$ and the image is transformed into a matrix of dimensionality $CR^2 \times N \left\lceil \frac{H-R}{S} + 1 \right\rceil \left\lceil \frac{W-R}{S} + 1 \right\rceil$.

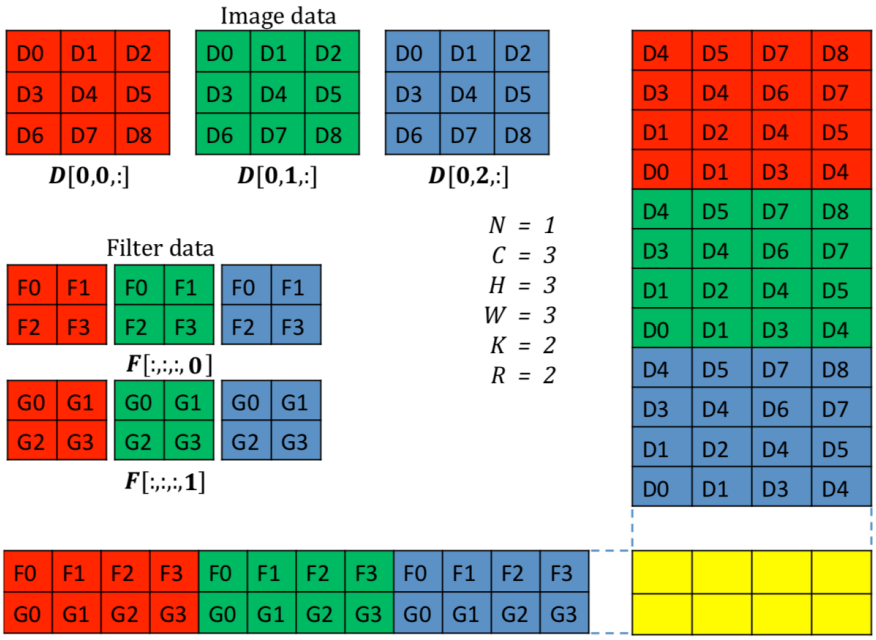

Figure 2 - Performing a convolution as a matrix-matrix multiplication. Image was modified from [12].

3 Silicon Photonics Background

3.1 Waveguides

Waveguides are manufactured on the silicon photonics platform by surrounding a silicon core with a silicon dioxide cladding. Since silicon has a high index of refraction, $n_{\mathrm{Si}} = 3.5$, compared to its oxide, $n_{\mathrm{SiO_2}} = 1.5$, the waveguide can be manufactured to have a width between 400 nm and 1000 nm while having a thickness of 220 nm. These waveguides have a bend radius of 5 µm and can support TE and TM polarized wavelengths between 1.5 µm and 1.6 µm [7]. Shows a scanning electron microscope image of a silicon waveguide and the TE mode propagation.

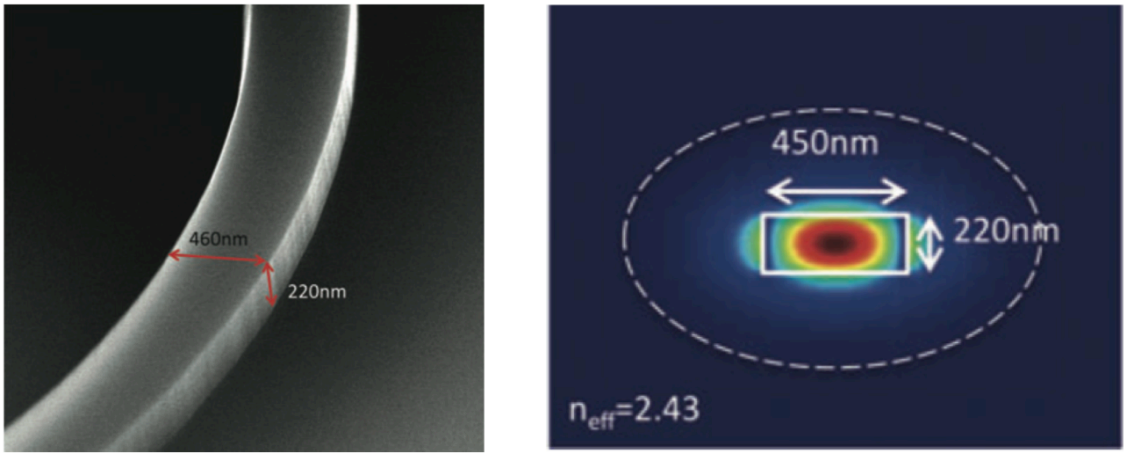

Figure 3 - Left: Scanning electron microscope image of silicon waveguide. Right: TE propagation of a similar waveguide. Both images taken from [13].

3.2 Microring Resonators

The ability for silicon waveguides to micrometer sized turning radiuses allows for the creation of microring resonators (MRRs). An MRR is a circular waveguide that is coupled with either one or two waveguides. The single waveguide configuration is called an all-pass MRR whereas the double waveguide configuration is called the add-drop resonator, as shown in Figure 4.

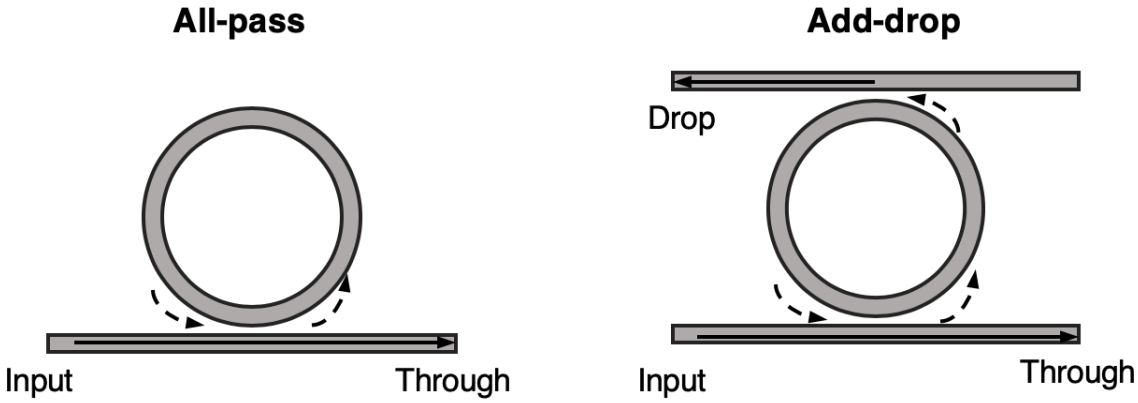

Figure 4 - All-pass and add-drop MRRs

The light from the waveguide is into transferred into the ring via directional coupler and then recombined. The effective index of refraction between the waveguide and the MRR and the

circumference of the MRR cause the recombined wave to have a phase shift, thereby interfering with the intensity of original light. Defining R as the radius of the ring and n_{eff} is the effective index of refraction between the ring and waveguide, the resonance frequencies of a particular MRR are [13]:

$$\lambda_r = \frac{2\pi R n_{eff}}{m}, \qquad m \in \mathbb{N}_{>0} \tag{5}$$

In order get the transmission of an MRR independent of wavelength, one defines the phase as:

$$\phi = \frac{4\pi^2 R n_{eff}}{\lambda} = 2\pi R n_{eff} \frac{\omega}{c} \tag{6}$$

The transfer function of the intensity of the light coming out through port with the light going into the input port of the all-pass resonator is described by:

$$T_n(\phi) = \frac{a^2 - 2ra\,cos(\phi) + r^2}{1 - 2ra\,cos(\phi) + (ra)^2} \tag{7}$$

where r is the self-coupling coefficient and a is the propagation loss from the ring and the directional coupler. For the add-drop MRR, the transfer function of the intensity of the through port light with respect to the input light is:

$$T_p(\phi) = \frac{(ra)^2 - 2r^2a\,cos(\phi) + r^2}{1 - 2r^2a\,cos(\phi) + (r^2a)^2} \tag{8}$$

and the transfer function of the intensity of the drop port light with respect to the input light is:

$$T_d(\phi) = \frac{(1-r)^2 a}{1 - 2r^2a\,cos(\phi) + (r^2a)^2} \tag{9}$$

In the case where the coupling losses are negligible, $a \approx 1$, the relationship between the add-drop through and drop transfer functions is:

$$T_p = T_d - 1 \tag{10}$$

Figure 5 demonstrates the Lorentzian line shape described by equations (9) and (10) in the domain $\phi \in [-\pi, \pi]$ where $a = 1$ and $r = 0.95$.

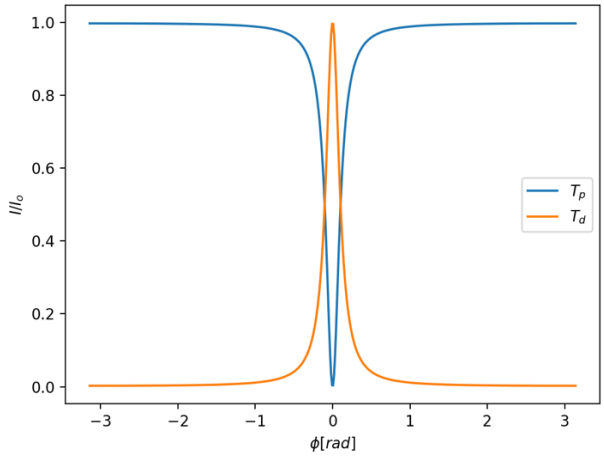

Figure 5 - Phase dependent transfer of MRR through port and drop port lines

3.2.1 Wavelength Division Multiplexing and Optical Modulation

Wavelength division multiplexing (WDM) is a technique where light of different wavelengths are travel through a single waveguide via an optical multiplexer. WDM also allows for multiplexed light travelling through a waveguide to be demultiplexed into several separate waveguides by wavelength. In silicon photonics, WDM is achieved by using add-drop MRRs [14]. Multiplexing using an MRR consists of coupling all of the light at MRR's resonant wavelength from the input port into the drop port. If multiple MRRs share the same drop port, then that waveguide will contain a set of multiplexed wavelengths. Similarly, demultiplexing is achieved by coupling light at a MRR's resonant frequency from a multiplexed waveguide into its own drop port.

If the MRR receiving a multiplexed signal in the input port is tuned slightly off resonance from a particular wavelength, only a portion of light at that wavelength will come out of the through port. Therefore, intensity modulation of multiplexed light is possible by

changing the resonant wavelength of an MRR. From equation (6), a change in the resonant wavelength is possible by a change the effective index of refraction. Applying heat across the coupling waveguide and ring changes effective index of refraction due to of the thermo-optic effect of silicon. By using ohmic heating, the amount of light coming out of the through ports of an MRR can be controlled via an analog electronic signal [15]. It was calculated that around ~100 multiplexed wavelengths between 1.5 μm and 1.6 μm can be modulated using a typical MRR [15].

4 Dot Products using Photonics

4.1 Overall Architecture

The fundamental operation of a convolution is the is dot product of two vectorized matrices. Therefore, one needs to understand how to compute a vector dot product using photonics before proposing an architecture of capable of performing convolutions.

A wavelength multiplexed signal consists of k electromagnetic waves, each with angular frequency ω_i. If it is assumed that each wave has an amplitude of E_0, a power enveloping function μ_i whose modulation frequency is significantly smaller than ω_i, then the slowly varying envelope approximation and a short-time Fourier transform can be used to derive an expression for the multiplexed signal in the frequency domain [15]:

$$E_{mux}(\omega) = \sum_{i=0}^{k-1} E_0 \sqrt{\mu_i}\, \delta(\omega - \omega_i) \tag{11}$$

In this model, $\delta(\omega - \omega_i)$ is the Dirac delta function and $\mu_i \geq 0$ since power envelopes cannot represent negative values. If the enveloping function is prevented from amplifying the electric field, μ_i can further restricted to the domain $0 \leq \mu_i \leq 1$. Next, given the tunable linear filters

$H^+(\omega)$ and $H^-(\omega)$ such that when the interact with the multiplexed fields, the following weighted signals are created:

$$E_w^-(\omega) = H^-(\omega)E_{mux}(\omega) \tag{12}$$

$$E_w^+(\omega) = H^+(\omega)E_{mux}(\omega) \tag{13}$$

Assuming that the two signals are fed into a balanced photodiode with spectral response $R(\omega)$, the induced photocurrent is described by:

$$i_{PD} = \int_{-\infty}^{\infty} d\omega\, R(\omega)(|E_w^+(\omega)|^2 - |E_w^-(\omega)|^2)$$

$$= \int_{-\infty}^{\infty} d\omega\, R(\omega)(|H^+(\omega)|^2 - |H^-(\omega)|^2)\,|E_{mux}(\omega)|^2$$

$$= \sum_{i=0}^{k-1} R(\omega_i)(|H^+(\omega_i)|^2 - |H^-(\omega_i)|^2)\,E_o r_i \tag{14}$$

Assuming that $R(\omega)$ is roughly constant in the area of spectral interest, one can set $x[i] = E_o R_o \mu_i$ and $w^*[i] = |H^+(\omega_i)|^2 - |H^-(\omega_i)|^2$ resulting in a photocurrent equal to:

$$i_{PD} = \sum_{i=0}^{k-1} x[i]w^*[i] = \boldsymbol{x} \cdot \boldsymbol{w}^* \tag{15}$$

The through ports and drop ports of an MRR can be used to implement the linear filters H^+ and H^- such that:

$$H^+ = T_d, \qquad T^- = T_d \tag{16}$$

Then, using equation (10) one can then represent the weights as:

$$w^*[i] = 2T_{d,i}(\phi_i) - 1 \tag{17}$$

Since ϕ_i is tuned to the resonance frequency ω_i equations (9) and (17) are used to get:

$$\phi_i = \arccos\left(-\frac{1}{2r^2a}\left(\frac{2(1-r)^2a}{w^*[i]+1} - 1 - (r^2a)^2\right)\right) \tag{18}$$

From equation (17), it can be seen that w_i^* can be between -1 and 1 since T_d is a passive filter that can only represent values between 0 and 1. In order perform a dot product with a weight vector $\boldsymbol{w}$ whose components are not limited to the range -1 to 1, a gain g to can be applied to the photocurrent such that:

$$\boldsymbol{x} \cdot \boldsymbol{w} = g\boldsymbol{x} \cdot \boldsymbol{w}^* = g \sum_{i=0}^{k-1} x[i]w^*[i] = \boldsymbol{x} \cdot \boldsymbol{w}^* \tag{19}$$

$$g = \max_{0 \le i \le k-1} |w[i]| \tag{20}$$

$$\boldsymbol{w} = g\boldsymbol{w}^* \tag{21}$$

assuming each ϕ_i corresponds to a weighting of w_i^*. This electronic gain can be performed using a transimpedance amplifier (TIA). A diagram of the electro-optic architecture described in this section is presented in Figure 6. From now on, this amalgamation of electronic and optical components is referred as a photonic weight bank (PWB). PWBs similar to the one in Figure 6 have been successfully implemented in the past [15, 16, 17].

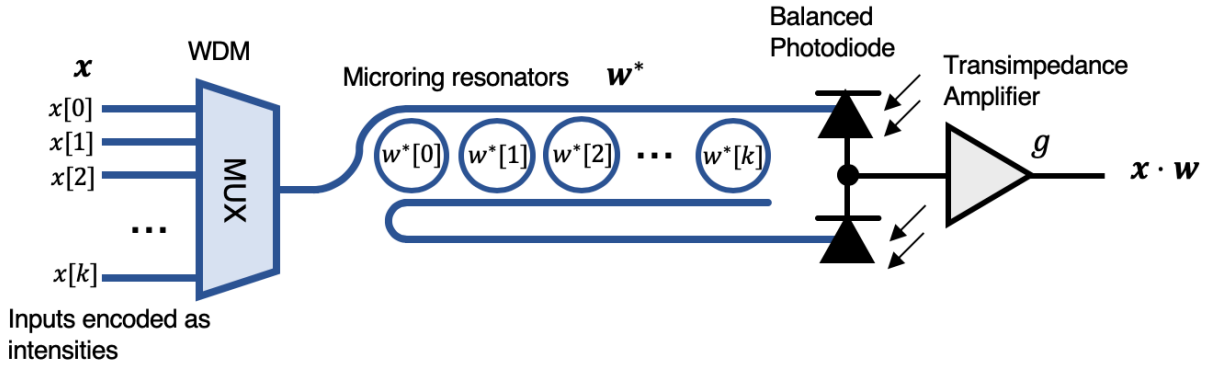

Figure 6 – An electro-optic architecture for performing dot products

4.2 Representing Negative Inputs

The formulation described in equations (11), (14) and (15) describes a simple normalized encoding scheme that allows $x[i]$ to represent values between 0 and 1. However, it is possible

to represent values between -1 and 1. This is achieved by modifying the power enveloping function to $\mu_i = \frac{1}{2}(x_i + 1)$. If the same set of derivations is followed with the same equation (19) is modified to:

$$g\left(\sum_{i=0}^{k-1} x[i]w^*[i] + \sum_{i=0}^{k-1} E_o R_o w^*[i]\right) \tag{22}$$

The second term in this sum is a predictable bias current term that conceptually be subtracted before feeding into the TIA. This is a disadvantage of supporting negative inputs, as additional optical or electronic control circuitry would need to be designed. Another trade-off is a loss in precision due to a larger range of inputs needing to be represented, analogous to the loss in precision with signed integers for classical computing.

4.3 Throughput and Size Estimation

The time takes for light to propagate from the MUX to before the PDs is:

$$t_{prop} = \frac{k2\pi r_{MRR}}{c} \tag{23}$$

Where c is speed of light $2\pi r_{MRR}$ is the circumference of the MRR and k is the number of MRRs. Assuming 100 MRRs with a radius of $10\ \mu m$ [15, 18], the PWB gets a propagation time of ~ 21 ps and an throughput of $\frac{1}{t_{prop}} = 50\ \mathrm{GS/s}$. The bottlenecks come from the balanced PDs has a throughput of $25\ \mathrm{GS/s}$ [19] and the TIA has a throughput of $10\ \mathrm{GS/s}$ [20]. An individual MRRs can be modulated at speeds of $128\ \mathrm{GS/s}$ [18], meaning that the modulation frequency of the MRRs does not bottleneck the throughput of the PWB.

5 Photonic Convolutions for CNNs

5.1 A Top-down View

The goal of this section is to present a photonic architecture capable of performing convolutions for CNNs. The architecture being proposed here takes inspiration from a previous approach called “PCNNA” [21], but goes a step further by allowing for greater parallelization. This new architecture is called DEAP, standing for “Digital Electronics and Analog Photonics”.

Figure 7 provides a high-level overview of the proposed architecture. For convenience, optical components are drawn with a blue outline and electronic components are drawn with a black outline. The idea is that one would represent the input values by modulating the intensities of a group of lasers with identical powers but unique wavelengths. These modulated inputs would be sent into an array of photonic weight banks which would then perform the convolution for each channel. Finally, the outputs of the weight banks would be summed using a voltage adder, which produces the convolved feature. The interfacing of optical components with electronics would be facilitated by the use of DACs and ADCs, while the storage of output and retrieving of inputs would be achieved by using GDDR SDRAM.

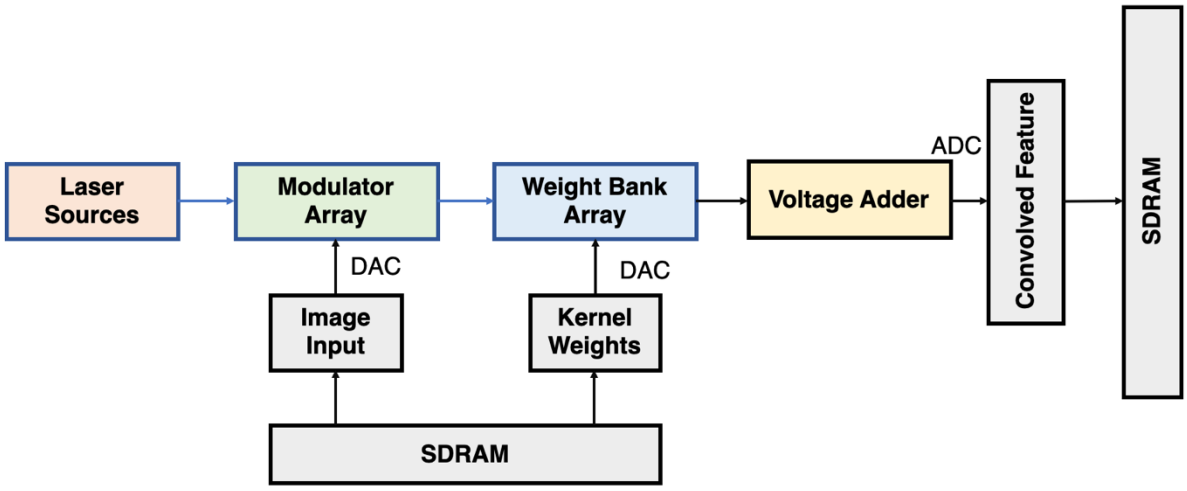

Figure 7 - Block diagram of DEAP architecture

In Table 5, we define some bounding parameters for DEAP. These bounding parameters represent the range of convolutional parameters that a particular implementation of DEAP can support. If a convolutional parameter described in Table 1 does not have a complementary bounding parameter, it means that the DEAP architecture can support for arbitrary values of said convolutional parameter.

Table 2 - DEAP bounding parameters

Parameter	Meaning
C_m	Maximum number of input channels.
R_m	Maximum kernel edge length

5.2 Producing a Single Convolved Pixel

For now, we only consider an architecture that can produce one convolved pixel at a time. To handle convolutions for kernels with dimensionalities up to $R_m \times R_m \times C_m$, we will require R_m^2 lasers with unique wavelengths since a particular convolved pixel can be represented as the dot product of two $1 \times R_m^2$ vectors. To represent the values of each pixel, we require $C_m R_m^2$ modulators (one per kernel value) where each modulator keeps the intensity

of the corresponding carrier wave proportional to the normalized input pixel value. Figure 8 shows what such an architecture would look like. The R_m^2 lasers are multiplexed together using WDM, which is then split into C_m separate lines. On every line, there are R_m^2 DAC-controlled all-pass MRRs, resulting in $C_m R_m^2$ MRRs in total. Each WDM line, L_k, will modulate the signals corresponding the a subset of R_m^2 pixels on channel k, meaning that the modulated wavelengths on a particular line correspond to the pixel inputs $\boldsymbol{D}[i{:}\, i + R_m, j{:} j + R_m, k]$ where $k \epsilon [0, C_m - 1]$.

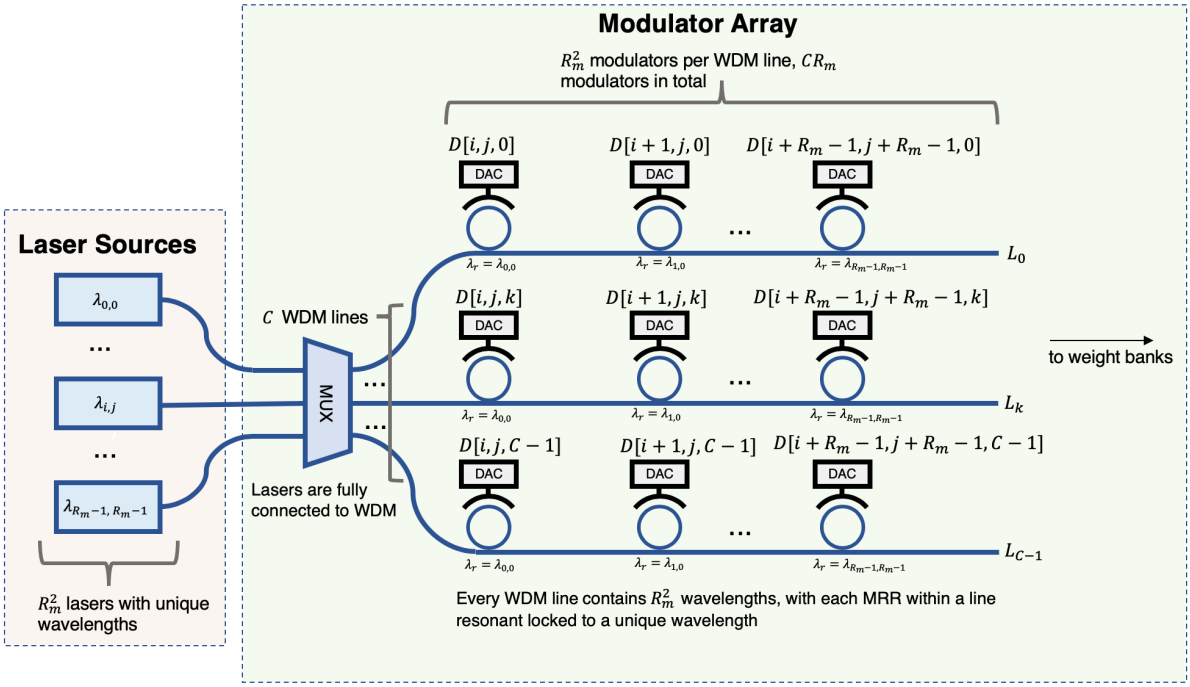

Figure 8 – DEAP input representation with photonics

A summary of the number of required components in presented in Table 3. The phase for an all-pass resonator corresponding to a particular intensity modulation value can be computed by using equation (7):

$$\phi_{k,l} = \arccos\left(-\frac{D[k,l](1+(ra)^2) - a^2 - r^2}{2ra(1-D[k,l])}\right) \tag{24}$$

resulting in a modulated input where,

$$D[k,l] = T_n(\phi_{k,l})|E_o|^2 \tag{25}$$

In order to represent negative inputs, the left-hand side of equation (25) can be set to $\frac{1}{2}(D[k,l]+1)$ as described in section 4.2.

Table 3 - Required components for DEAP input representation

Component	Required Number
Number of Wavelengths	R_m^2
MRR modulators	$C_m R_m^2$
WDM Lines	C_m

The C_m WDM lines will then be fed into an array of C_m PWBs. Each PWB will contain R_m^2 MRRs with the weights corresponding to the kernel values at a particular channel. For example, WB_k should contain the vectorized weights for the kernel $\boldsymbol{F}[:,:,k]$. Each MRR within a PWB should be tuned unique the resonant wavelength within the multiplexed signal. This architecture is shown in Figure 9, with the summary of parameters in Table 5. The outputs of the weight bank array are C_m electrical signals, each proportional to the dot product $\boldsymbol{F}[:R_m,:R_m,k] \cdot \boldsymbol{D}[:,:,k]$. To perform a convolution with a kernel edge length less than R_m, one can set $\boldsymbol{F}[R+1:,R+1]$ to zero.

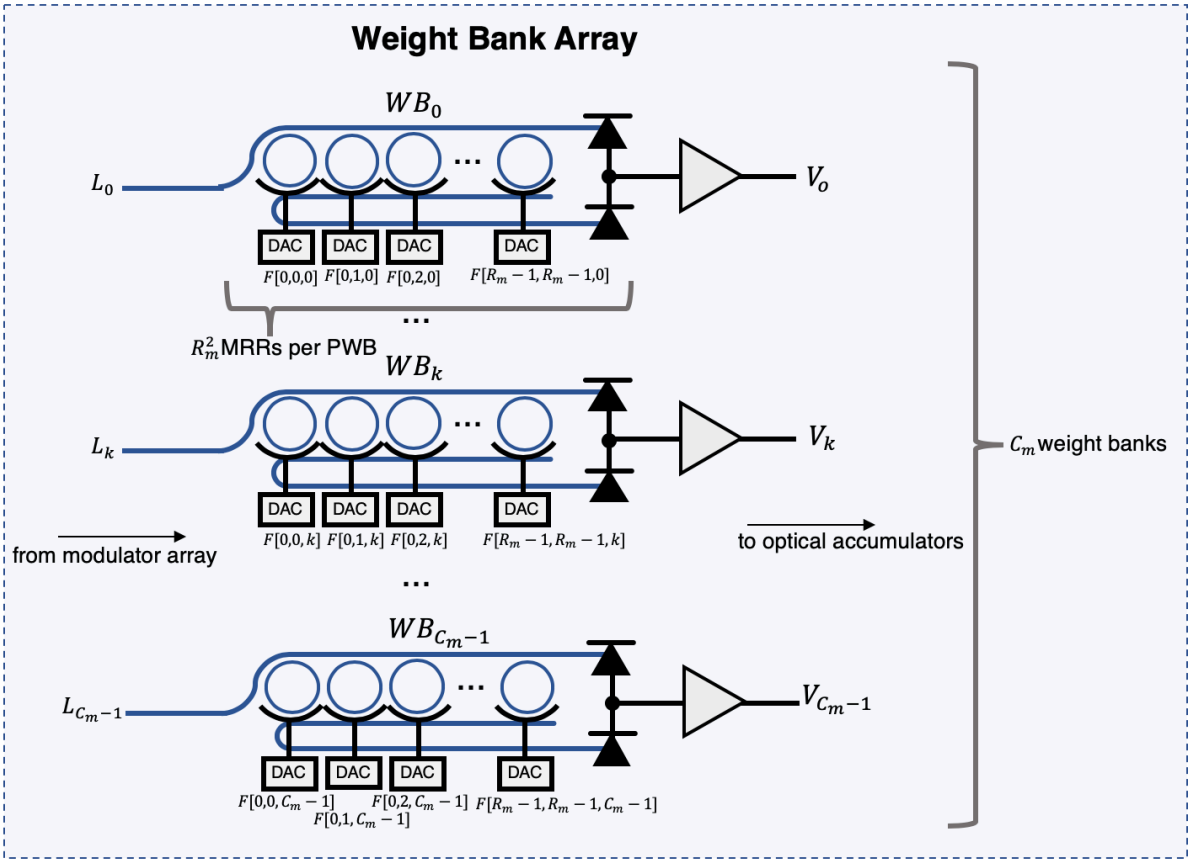

Figure 9 – Applying the kernel with photonics for DEAP

Table 4 – DEAP required components for applying kernel

Component	Required Number
Photonic Weight Banks	C_m
MRR modulators per PWB	R_m^2

In order to get to equation (4), the signals from the weight banks need to be added together. This can be achieved using a passive voltage adder, as shown in Figure 10. The output from this adder will therefore be the value of a single convolved pixel which will be converted into a digital signal using an ADC.

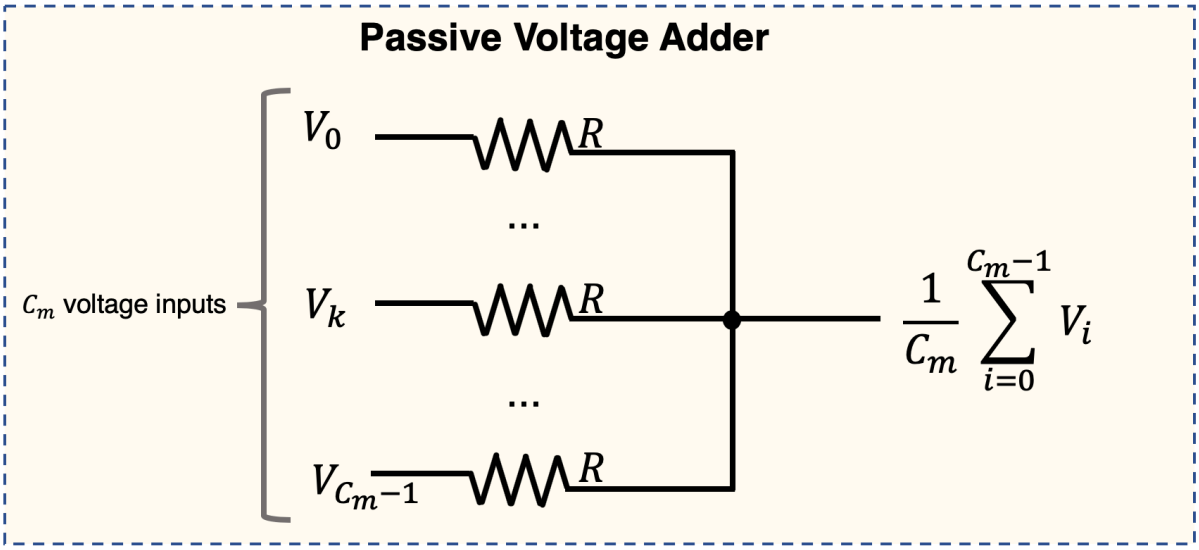

Figure 10 – Passive voltage adder design

5.3 Performing a Full Convolution

In the previous section, we have discussed how DEAP can produce a single convolved pixel. In order to perform a convolution of arbitrary size, one would need to stride along the input image and readjust the modulation array. Since the same kernel is applied across the set of inputs, the weight banks do not need to be modified until a new kernel is applied. Figure 11 demonstrates this process on an input with $S = 1$. To handle $S > 1$, the inputs being passed in to DEAP should also be strode accordingly. In this approach, the inputs should have been zero padded before being passed into DEAP. In pseudocode, performing a convolution with K filters can be implemented as shown in Algorithm 1.

```
Algorithm 1 – Convolutions for CNNs using DEAP
function convolve(D, F, R, O, S, H, W) do
    for (k = 0; k < K; k = k + 1) do
        load kernel weights from F[:, :, :, k]
        for (h = 0; h < H – R + 1; h = h + S) do
            for (w = 0; w < W – R + 1; w = w + S) do
                load inputs from D[h:min(h+R, H), w:min(w+R, W), :]
                perform convolution
                store results in O[h/S, w/S, k]
            end
        end
    end
end
```

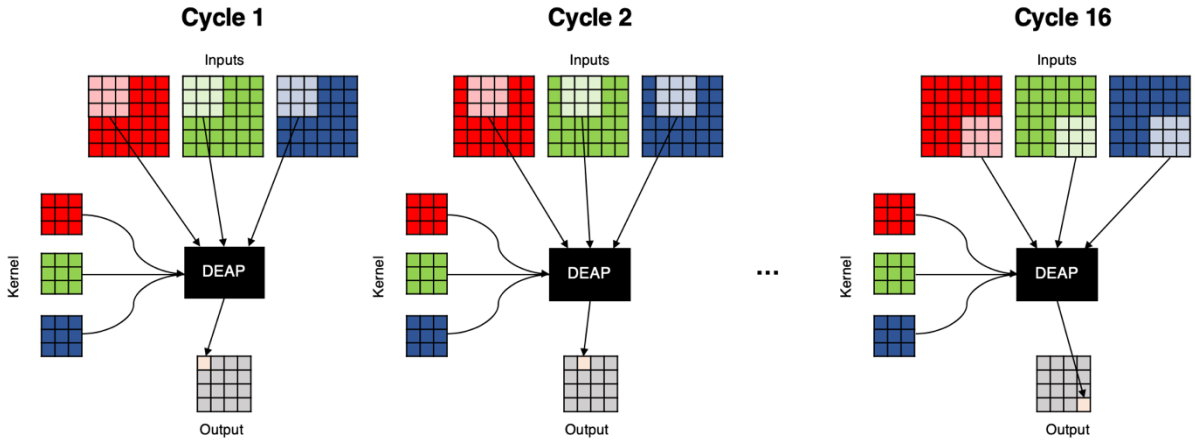

Figure 11 - Performing a convolution using DEAP

The DEAP architecture also allows for parallelization by treating the photonic architecture proposed in the previous section as a single output "convolutional unit". However, by creating n_{conv} instances of these convolutional units, you could produce n_{conv} pixels per cycle by passing in the next set of inputs per unit. This is demonstrated in Figure 12 for $n_{conv} = 2$. The computation of output pixels can be distributed across each convolutional unit, resulting in a runtime complexity of $O\left(\frac{KHW}{S^2 n_{conv}}\right)$.

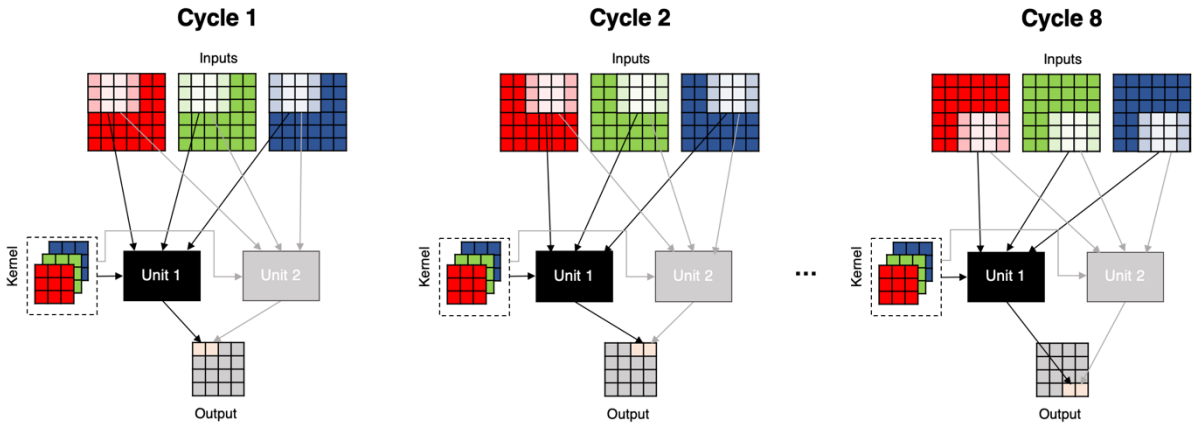

Figure 12 - Performing a convolution with two convolutional units

5.4 Throughput and Energy Estimation

As discussed in section 4.3, the throughput of a PWB is around 5 GS/s. The DACs [22] and ADCs both [23] operate at 5 GS/s and support to 7-bits. The GDDR6 SDRAM operates at 16 G with a 256-bit bus size [24]. Consequently, the speed of the system is limited by the throughput of the DACs/ADCs, resulting in DEAP producing a single convolved pixel at 5 GS/s or $t = 200$ ps.

The energy used by a single DEAP convolutional unit depends on the R_m and C_m parameters. The 100-wavelength limitation for MRRs constrains the maximum R_m to be 10 as each multiplexed waveguide will store R_m^2 signals. The number of MRRs used in the modulator array is equal to $R_m^2 C_m$, meaning that only certain C_m and R_m^2 values are allowed for a finite number of MRRs. Assuming that a maximum of 1024 MRRs can be manufactured in the modulator array, allowing a convolutional unit to support a large kernel size with a limited number of channels, $R_m = 10$, $C_m = 12$, or a small kernel size with a large number of channels, $R_m = 3$, $C_m = 113$. We will consider both edge cases to get a range of energy consumption values. For the smaller convolution size, we will have R_m^2 lasers, $R_m^2 C_m$ MRRs

and DACs in the modulator array, $R_m^2 C_m$ MRRs and C_m TIAs in the weight bank array and one ADC to convert back into digital signal. With 100 mW per laser, 19.5 mW per MRR, 26 mW per DAC, 17 mW per TIA [20] and 76 mW per ADC, we get an energy usage of 112W for the large kernel size and 95W for the smaller kernel size. Therefore, we estimate a single convolution unit to use around ~100W when 1024 modulators are used to represent inputs.

6 Photonic Convolutions for General Image Processing

6.1 Rationale for a second approach

In section 5, an architecture for performing convolutions was proposed. This architecture is optimized for convolutional neural networks in which the number of channels for an image can reach an arbitrary value. In image processing, convolutions are only applied to a single channel with stride always equal to 1. In this section, we propose an alternate photonic architecture, "DEAP-GIP", that is specifically optimized for general image processing (GIP) workloads. Where DEAP's parallelization abilities are on channel-level matrix dot products, DEAP-GIP is parallelized by allowing for an entire convolution to happen in one step. We introduce additional bounding parameters described for this approach in Table 5.

Table 5 – DEAP-GIP bounding parameters

Parameter	Meaning
H_m	Maximum input image height
W_m	Maximum input image height
R_l	Minimum kernel edge length

6.2 Input Representation

As before, to handle convolutions for kernels with dimensionalities up to $R_m \times R_m \times 1$, we will require R_m^2 lasers with unique wavelengths. The assignment of lasers wavelengths to pixel indexes is done such that the pixel $D[i,j]$ will use $\lambda_{i \,\%\, R_m, j \,\%\, R_m}$ as its carrier wavelength. In the case where $H_m \,\%\, R_m = 0$ and $W_m \,\%\, R_m = 0$, each wavelength is shared an equal number of times. Otherwise, asymmetric wavelength assignment occurs, in which some wavelengths will get shared more than others. Though DEAP-GIP can support asymmetric wavelength assignment, manufacturing is simpler when the bounding parameters are chosen such that $H_m \,\%\, R_m = 0$ and $W_m \,\%\, R_m = 0$. For this reason, the rest of the section assumes that symmetric wavelength assignment is used. Figure 13 demonstrates what the wavelength assignment should look like, where each cell represents an input pixel. Note how no matter which $R \times R,\ R \leq R_m$ submatrix is picked, each element will have a unique wavelength assigned to it.

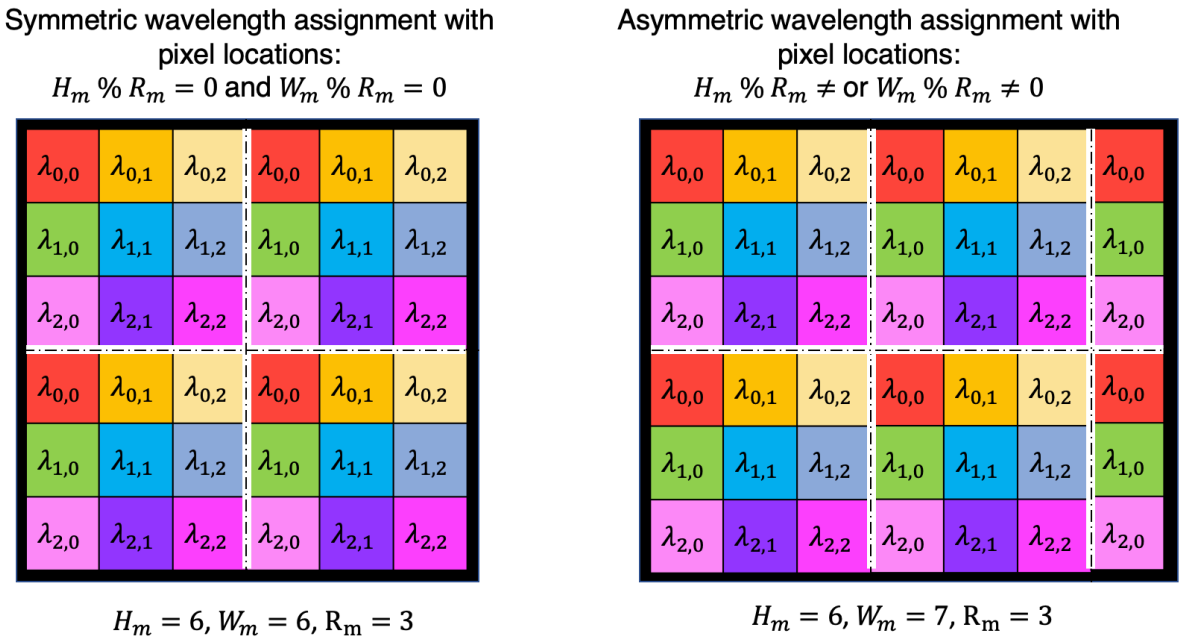

Figure 13 - Wavelength assignment to pixel locations

To represent the values of each pixel, $H_m \times W_m$ modulators are required to change the intensity of the corresponding carrier waves. Figure 14 shows what such an architecture would look like.

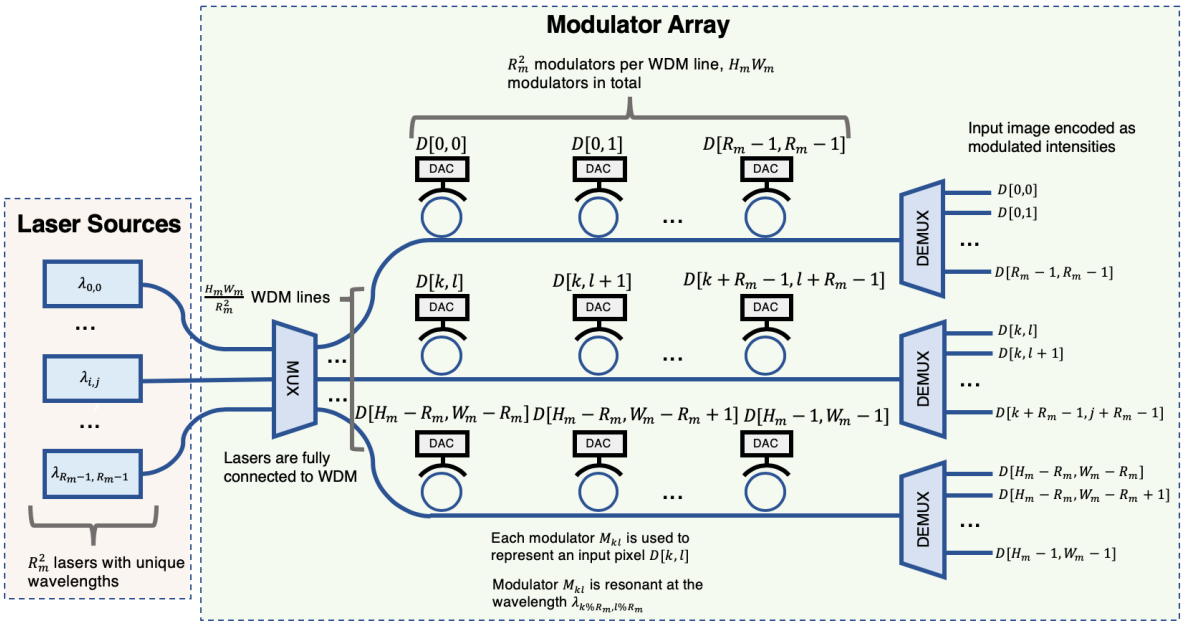

Figure 14 - DEAP-GIP input representation with photonics

The R_m^2 lasers are multiplexed together, which is then split into $H_m W_m / R_m^2$ separate lines. On every line, there are R_m^2 all-pass MRRs, resulting in $H_m W_m$ MRRs in total. Modulator, $M_{k,l}$, is set to correspond to the pixel value located at $D[k,l]$ and is tuned to the wavelength $\lambda_{k\ \%\ R_m, l\ \%\ R_m}$. Each WDM line, $L_{p,q}$, contain the modulated the signals corresponding to $D[p * H_m : p * H_m + R_m, q * W_m : q * W_m + R_m]$ where $p \in [0, H_m/R_m - 1]$ and $q \in [0, W_m/R_m - 1]$. All of these lines are sent through a demultiplexer, resulting in $H_m W_m$ output signals, each corresponding to an input pixel. A summary of the number of required components in presented in Table 6.

Table 6 - Required components for DEAP-GIP input representation

Component	Required Number
Number of Wavelengths	R_m^2
MRR modulators	$H_m W_m$
WDM Lines	$H_m W_m / R_m^2$

6.3 Kernel Representation

If we only want to support one kernel size, we could have $(H_m - R_m + 1)(W_m - R_m + 1)$ photonic weight banks where weight bank $W_{m,n}$ is connected to the inputs $D[m{:}\, m + R_m, n{:}\, n + R_m]$ for the range $m \epsilon [0, H_m - R_m - 1]$ and $n \epsilon [0, W_m - R_m - 1]$. Supporting smaller kernel sizes down to R_l would require $(H_m - R_l + 1)(W_m - R_l + 1)$ weight banks. This changes the assignment of weight banks to inputs such that $W_{m,n}$ is connected to the inputs $D[m{:}\, m + q, n{:}\, n + q]$ where $q = \min(R_m,\ \min(H_m - m, W_m - n))$ given the new range $m \epsilon [0, H_m - R_l - 1]$ and $n \epsilon [0, W_m - R_l - 1]$. Figure 15 give a visual representation of assigning inputs to weight banks.

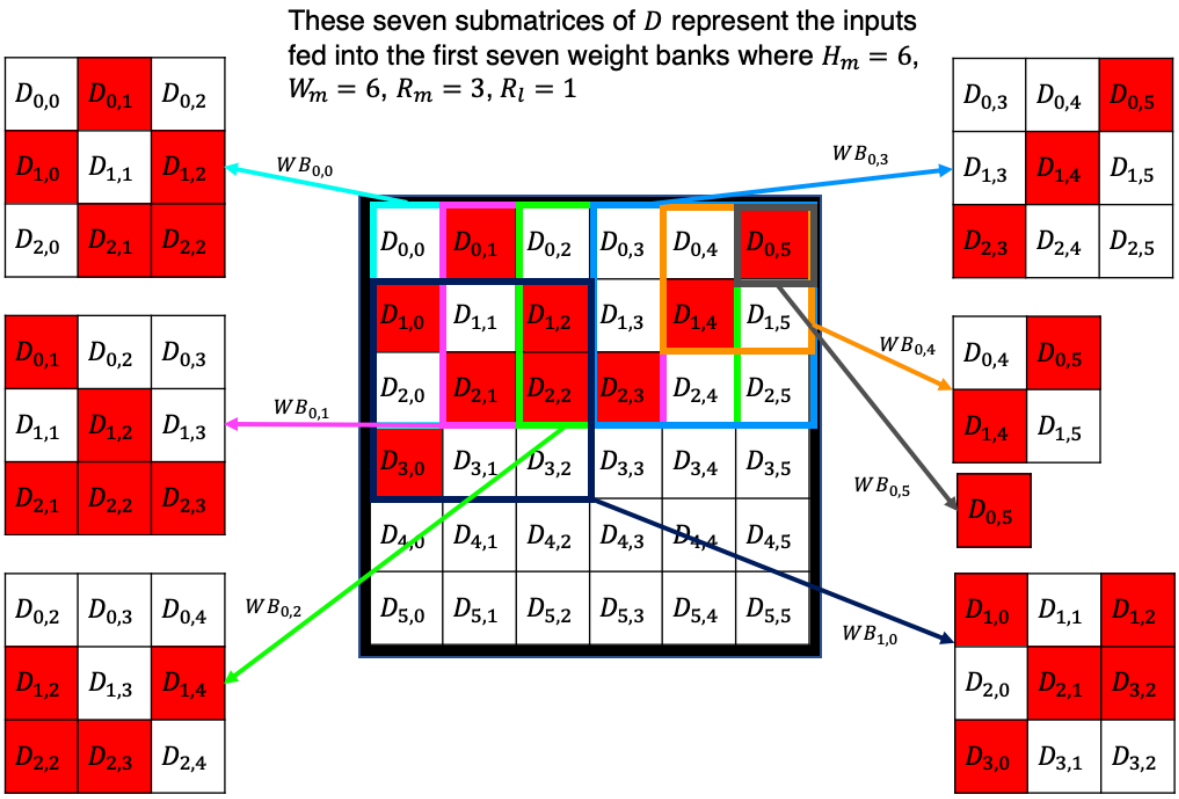

Figure 15 - DEAP-GIP input assignment to weight banks

Each photonic weight bank will consist of a number of MRRs between R_l^2 and R_m^2 depending on their input connections. The weighting of each MRR should correspond to a value of a kernel, such that for a particular $R, R_l \leq R \leq R_m$, the weights $w[:R, :R]$ will corresponds to the kernel values $F[:,:]$ and $w[R:,R:] = 0$. If a weight bank has fewer input connections than R, then all of the weights should be set to 0. Following this approach allows us to change the size of the filter size without needing to design a whole new hardware implementation. This means that each PWB computes equation (4) for one particular output pixel. By having multiple weight banks, all of the output pixels can be computed in a parallel, at once. Figure 16 presents a photonic architecture for applying the kernel.

It should be noted that the outputs of the demultiplexer in Figure 14 will be shared across multiple weight banks, meaning the waveguides will need to be split. Each output waveguide from the DEMUXs in Figure 14 will be shared by up to R_m^2 PWBs at a time. Since each waveguide is split $H_m W_m / R_m^2$ times during the first WDM, the intensity of a given laser will be attenuated by a factor of $1/H_m W_m$. Assuming that the lasers operate at 100 mW, the power incident to a photodiode is equal to -4 dBm in the extreme case of $H_m = W_m = 1000$. Considering that photodiodes are sensitive to intensities down to -25 dBm, the reduction in input power will not be an issue. Another challenge with this approach is that manufacturing becomes more complicated as the waveguides need to be designed such that they do not intersect each other, though there has been some progress in manufacturing multi-planar waveguides using silicon photonics [25].

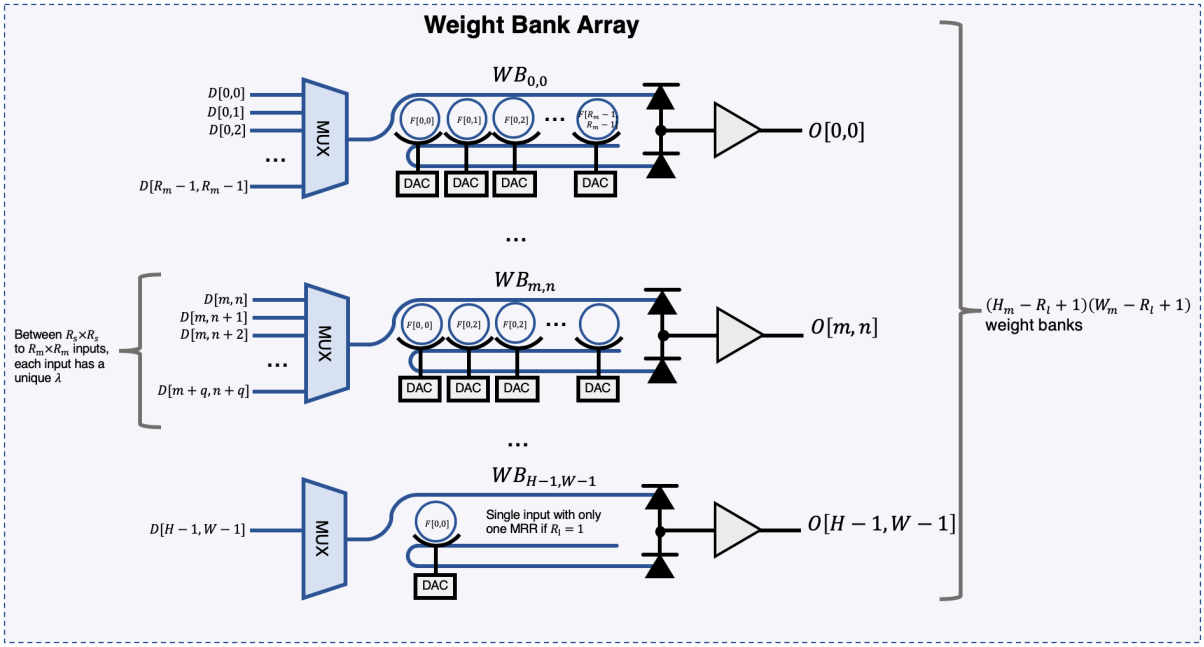

Figure 16 – Applying the kernel for DEAP-GIP

A summary of the number of required components is presented in Table 7

Table 7 - DEAP-GIP required components for applying kernel

Component	Required Number
Photonic Weight Banks	$(H_m - R_l + 1)(W_m - R_l + 1)$
MRR modulators per PWB	Between R_l^2 and R_m^2
Inputs to PWB	Between R_l^2 and R_m^2

6.4 Performing a Convolution

Although DEAP-GIP has a more complicated hardware architecture than DEAP and is constrained for convolutions with $S = 1$ and $C = 1$, DEAP-GIP allows for massive parallelization for image processing workloads in which the same filter is applied to a large number of images. If image size is less than or equal to $H_m \times W_m$ all the inputs can be loaded into DEAP-GIP and the entire convolved feature will be produced. If the image size is greater than $H_m \times W_m$, then the inputs can be loaded into the system as described in Algorithm 2. Figure 17 shows an example of a convolution on one input image where $H_m = W_m = 8$ but

$H = 8$ and $W = 14$. Note how the massive parallelization abilities of DEAP-GIP allow the entire convolution to be performed using only two input cycles. This is due to the runtime complexity of DEAP-GIP being $O\left(\frac{(H-R+1)(W-R+1)}{(H_m-R+1)(W_m-R+1)}N\right)$.

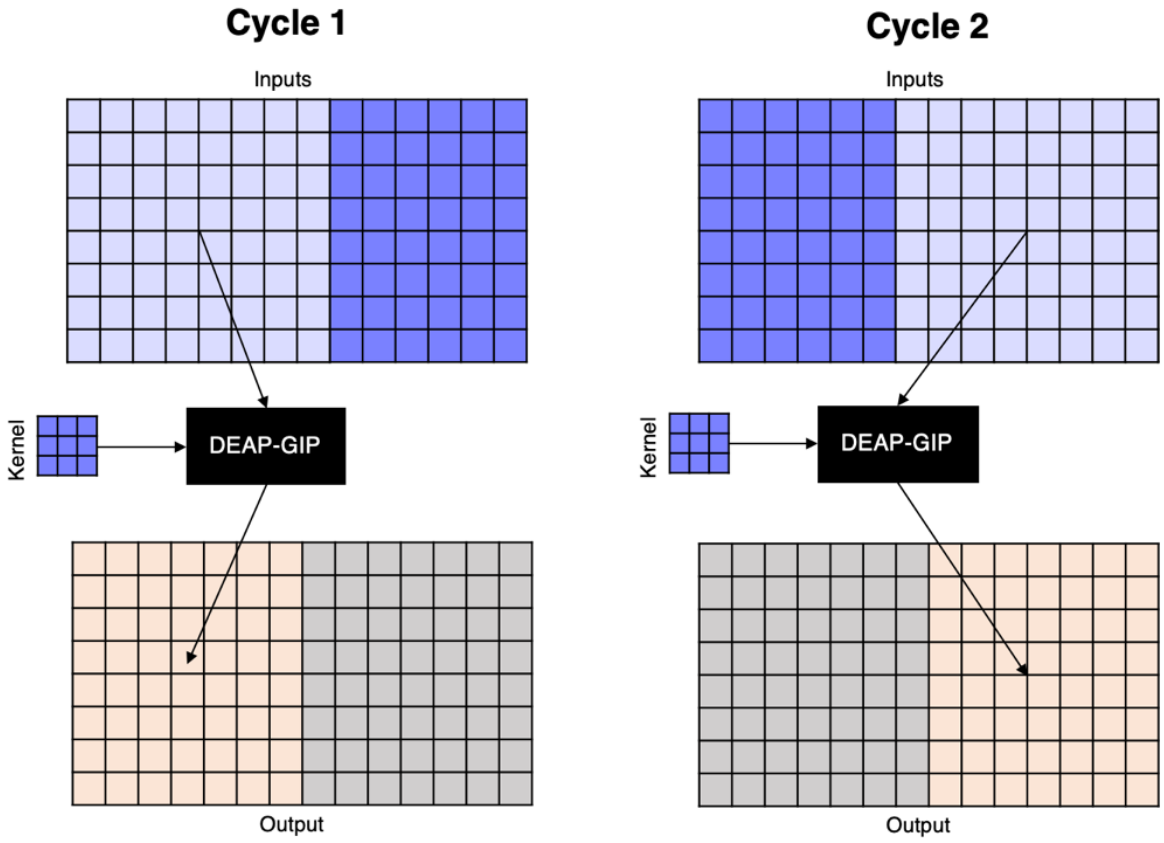

Figure 17 - Performing a convolution using DEAP-GIP

```
Algorithm 2 – Convolutions for Image Processing using DEAP-GIP
function convolve(D, F, R, O, N, H, W, Hm, Wm) do
    load kernel weights from F[:, :, 0]
    let Hout = H − R + 1
    let Wout = W − R = 1
    for (n = 0; n < N; n = n + 1) do
        for (h = 0; h < H − R + 1; h = h + Hm − R + 1) do
            for (w = 0; w < W − R + 1; w = w + Wm − R + 1) do
                load inputs from
                    D[h:min(h + Hm, H), w:min(w + Wm, W), 0, n]
                perform convolution
                store results in
                    O[h:min(h + Hm, H) − R + 1,
                      w:min(w + Wm, W) − R + 1, 0, n]
            end
        end
    end
end
```

6.5 Throughput and Energy Estimation

DEAP-GIP uses the same components as DEAP, meaning that it has the same throughput of $5GS/s$ assuming that all the input weights can be loaded in between the DACs clock cycles. Even though both approaches have the same throughput in terms of "samples", DEAP-GIP is more performant because a "sample" refers to a larger domain of output pixels.

DEAP-GIP requires R_m^2 lasers, $H_m W_m$ MRRs and DACs in the modulator array, an upper bound of $R_m^2(H_m - R_l - 1)(W_m - R_l + 1)$ MRRs and DACs in the weight bank array and $(H_m - R_l - 1)(W_m - R_l + 1)$ TIAs and ADCs. The power consumption of the components are 100 mW per laser, 19.5 mW per MRR, 26 mW per DAC, 17 mW per TIA and 76mW per ADC. With the assumption that $H_m = W_m = 12$, $R_l = 3$ and $R_m = 7$ we get a power consumption of 690 W.

7 Benchmarks

7.1 Proof of Concept Simulation

A high-level software simulator for DEAP and DEAP-GIP was developed. This simulator works by using the transfer function of the MRRs, through port and drop port summing equations at the balanced PDs and the TIA gain term to simulate a convolution. The simulator assumes that the MRRs can only be controlled with 7-bits of precision as that has been empirically observed in a lab setting. The simulations assume that the MRR self-coupling coefficient is equal almost 1 and equal to the loss, $r = a = 0.99$ [13]. The software does not attempt to model any sort of higher order errors due to shot, Johnson-Nyquist and flicker noise. Figure 18 shows an example of a gaussian blur convolution using Algorithm 1 for DEAP and Algorithm 2 for DEAP-GIP. The "convolve2d" [26] function in SciPy was used as the reference convolution implementation. The difference between DEAP and SciPy was quantified by taking the mean squared error (MSE) of each of the output pixels. The source code for the simulation is included Appendix A.

Figure 18 - DEAP and DEAP-GIP Simulation Results

7.2 Estimated DEAP Performance

DeepBench [27] is an empirical data set that contains how long various types of GPUs took to perform a convolution for a given set of convolutional parameters. The runtime of three of convolutional benchmarks for the GPUs presented in Table 8 were taken from the DeepBench dataset. Appendix A.1 contains the parameters used for each of these benchmarks.

Table 8 - Benchmarked GPUs with power consumption

GPU	Power usage
AMD Vega FE	375 W [28]
AMD MI25	300 W [29]
Nvidia Tesla P100	250 W [30]
Nvidia GTX 1080Ti	250 W [31]

Knowing that one DEAP convolutional unit can produce a pixel in 200 ps, an estimation for the runtime can be computed using equation (26):

$$t_{runtime} = 200\ \text{ps} * \frac{NK}{n_{conv}}\left(\frac{H-R}{S}+1\right)\left(\frac{W-R}{S}+1\right) \tag{26}$$

The estimated DEAP runtimes using one and two convolutional units were plotted against actual DeepBench runtimes in Figure 19. From this, we can see that using two convolutional unit performs slightly better than all the GPU benchmarks. While mean GPUs power consumption is 295 W, DEAP with a single convolutional unit uses about 110 W. Therefore, DEAP can perform convolutions between 1.4 and 7.0 times faster than the mean GPU runtime while using 0.37 times the energy consumption. Using two convolutional units doubles the speed of DEAP, meaning that DEAP can be between 2.8 and 14 times faster than a conventual GPU while using almost 0.75 times the energy consumption. DEAP with a single unit performing at a speed somewhat similar to the GPUs is expected.

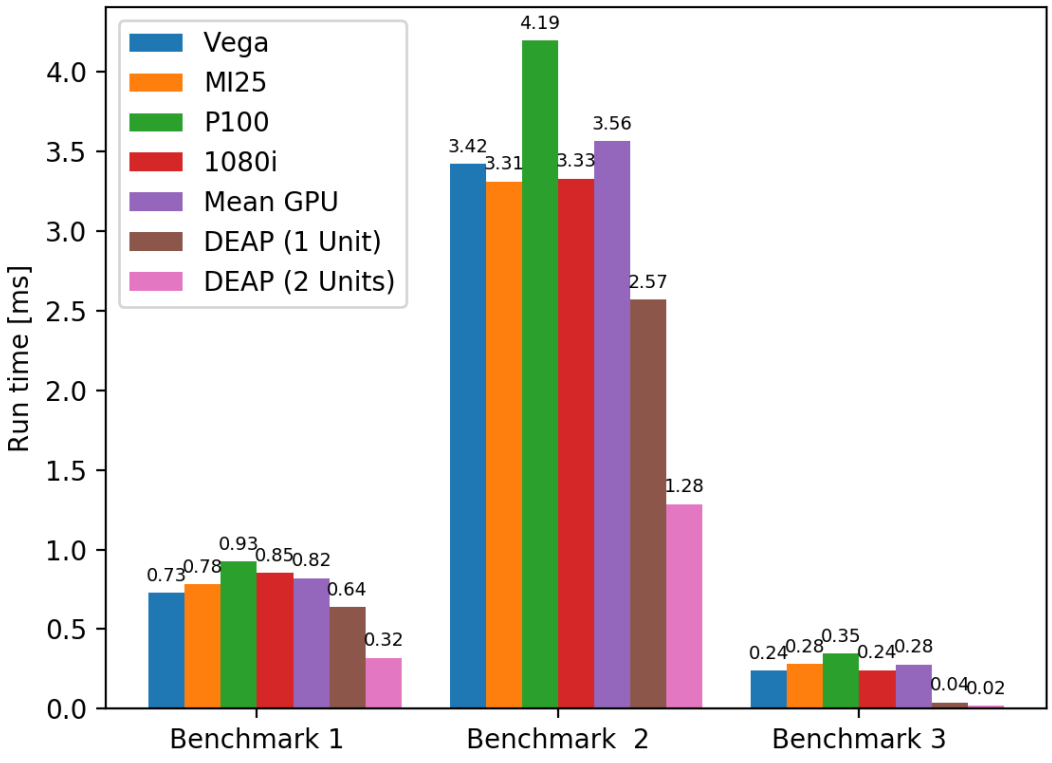

Figure 19 - Estimated DEAP convolutional runtime compared to actual GPU runtimes from DeepBench benchmarks

7.3 Estimated DEAP-GIP Performance

The same DeepBench dataset with the GPUs in Table 8 was used to provide a benchmark for DEAP-GIP. In this case, only the benchmarks compatible with the implementation described in section 6.5, meaning that $S = 1$, $K = 1$ and $3 \leq R \leq 7$. To estimate the runtime of DEAP-GIP for a set of convolutional parameters, equation (27) was used.

$$t_{runtime} = 200 \text{ ps} * \frac{(H - R + 1)(W - R + 1)}{(H_m - R + 1)(W_m - R + 1)} NK \tag{27}$$

A plot of the DEAP-GIP's estimated runtime against the actual DeepBench runtimes is presented in Figure 20 for these two particular benchmarks, DEAP-GIP has an estimated speedup of 4.5 and 68 times compared to the mean GPU performance. DEAP-GIP uses around 690 W of power, meaning that the increased speed comes at the expense of around 2.3 times more energy consumption.

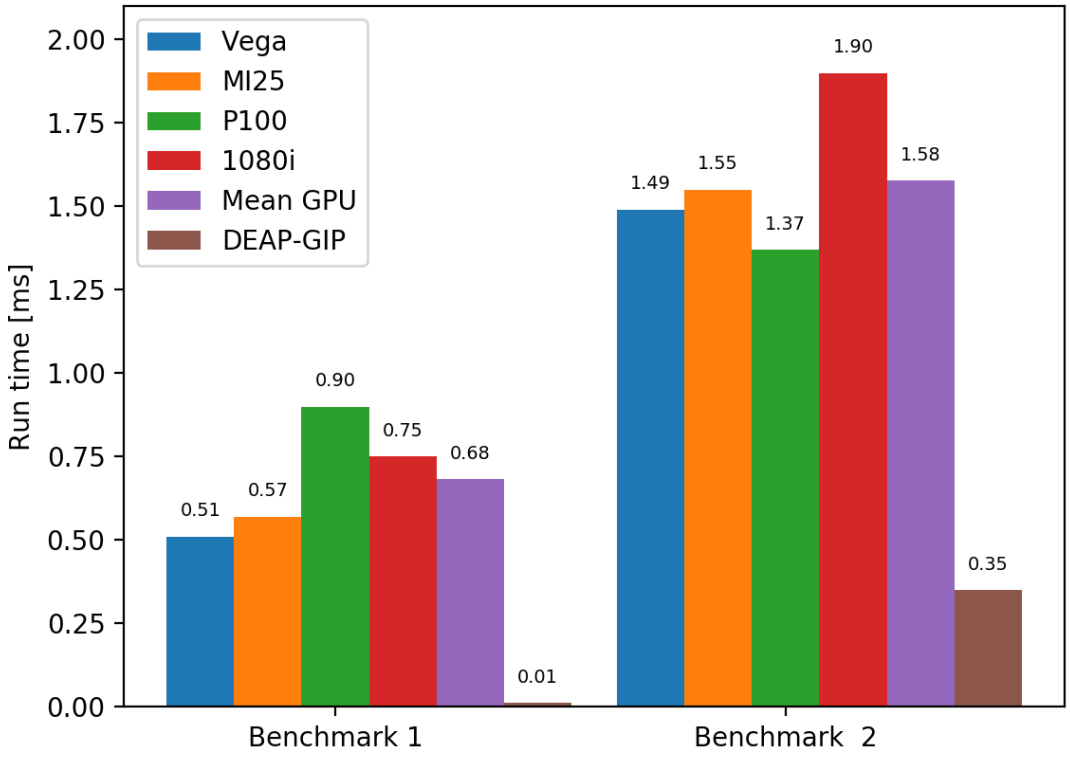

Figure 20 - Estimated DEAP-GIP convolutional runtime compared to actual GPU runtimes from DeepBench benchmarks

8 Conclusion

In this thesis, two photonic architectures for performing convolutions were proposed. The first approach, DEAP, is better suited for convolutional neural networks whereas the second, DEAP-GIP, is suited for general image processing. DEAP was estimated to be up 14 times faster than a GPU for convolutions while using almost half the energy consumption. A linear increase in processing speeds corresponds to a linear increase in energy consumption, making DEAP scalable. DEAP-GIP, which can only be run on single channel inputs with a convolutional stride of one, was estimated to be up 68 times faster than a GPU while using 3 times more energy consumption. High level software simulations have shown that both DEAP and DEAP-GIP are theoretically capable of performing a convolution. The largest bottleneck to both these systems came from the I/O interfacing of the optical components to the digital

ones. While photonic DACs [1] and ADCs [2] have been built the past, these implementation have bit precisions too low for convolutions (< 3 bits). If higher bit precision photonic DACs and ADCs are able to be built, replacing the electronic components with optical ones can significantly decrease the runtime.

In order to build these architectures in real life, there are a number of issues that still need to be solved. For one, electronically controlling the gain of the TIA requires the design of a voltage-controlled resistor, which can be implemented using field effect transistors [32]. There also needs to be control circuitry that routes the outputs of the SDRAM into the relevant DACs and from the ADCs into the SDRAM. Another issue is that DEAP and DEAP-GIP both process their data as analog signals, whereas GPUs perform floating point arithmetic. Though floating-point arithmetic does have some degree of error due to rounding in the mantissa, their errors are deterministic and predictable. On the other hand, the errors from photonics are due to stochastic shot, Johnson-Nyquist and flicker noises. However, artificially adding random noise to CNNs have been shown to reduce over-fitting [33], meaning that some degree of stochastic behaviour is tolerable in the domain of machine learning problems. Finally, MRRs have only been shown to have up to 7-bits of precision, which is significantly smaller than the range precision supported by even half-precision (16-bit) floating point representations. In conclusion, photonics have the potential to perform convolutions at speeds faster than top-of-the-line GPUs while having a lower energy consumption. Moving forward, the greatest challenges to overcome have to do with increasing the precision of photonic components so that they are comparable to classical floating-point representations.

Main body word count: 5600

9 References

[1] F. Zhang, B. Gao, X. Ge and S. Pan, "Simplified 2-bit photonic digital-to- analog conversion unit based on polarization multiplexing," *Optical Engineering,* vol. 55, no. 3, p. 031115, 2016.

[2] S. Pantoja, M. A. Piqueras, P. Villalba, B. Martínez and E. Rico, "High performance photonic ADC for space applications," in *International Conference on Space Optics — ICSO 2010*, 2017.

[3] J. Canny, "A Computational Approach To Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vols. PAMI-8, no. 6, pp. 679-698, 1986.

[4] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv:1506.02640 [cs.CV],* 2015.

[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs.CV],* 2015.

[6] X. Li, G. Zhang, H. H. Huang, Z. Wang and W. Zheng, "Performance Analysis of GPU-based Convolutional Neural Networks," in *45th International Conference on Parallel Processing*, Philadelphia, 2016.

[7] A. Rahim, T. Spuesens, R. Baets and W. Bogaerts, "Open-Access Silicon Photonics: Current Status and Emerging Initiatives," *Proceedings of the IEEE,* vol. 106, no. 12, pp. 2313-2330, 2018.

[8] M. A. Nahmias, B. J. Shastri, A. N. Tait, T. F. de Lima and P. R. Prucnal, "Neurmorphic Photonics," *Optics and Photonics News,* 2018.

[9] K. O'Shea and R. Nash, "An Introduction to Convolutional Neural Networks," *arXiv:1511.08458v2 [cs.NE],* 2015.

[10] K. Mehrotra, C. K. Mohan and S. Ranka, Elements of Artificial Neural Networks, Cambridge: MIT Press, 1997.

[11] G. Tan, L. Li, S. Triechle, E. Phillips, Y. Bao and N. Sun, "Fast implementation of DGEMM on Fermi GPU," in *SC '11: Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, Seatle, 2011.

[12] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, "cuDNN: Efficient Primitives for Deep Learning," *arXiv:1410.0759v3 [cs.NE],* 2014.

[13] W. Boegarts et al., "Silicon microring resonators," *Laser Photonics Rev.,* vol. 6, no. 1, p. 47–73 , 2012.

[14] M. Lipson, "Guiding, Modulating, and Emitting Light on Silicon—Challenges and Opportunities," *Journal of Lightwave Technology,* vol. 23, no. 12, pp. 4222-4238, 2005.

[15] A. N. Tait et al., "Microring Weight Banks," *IEEE Journal of Selected Topics in Quantum Electronics,* vol. 22, no. 6, 2016.

[16] A. N. Tait, T. F. Lima, M. A. Nahmias, B. J. Shastri and P. R. Prucnal, "Multi-channel control for microring weight banks," *Optics Express,* vol. 24, no. 8, pp. 8895-8906, 2016.

[17] A. N. Tait et al., "Feedback control for microring weight banks," *Optics Express,* vol. 26, no. 20, pp. 26422-26443, 2018.

[18] J. Sun, R. Kumar, M. Sakib, J. B. Driscoll, H. Jayatilleka and H. Rong, "A 128 Gb/s PAM4 Silicon Microring Modulator WithIntegrated Thermo-Optic Resonance Tuning," *Journal of Lightwave Technology,* vol. 37, no. 1, 2019.

[19] Z. Huang et al., "25 Gbps low-voltage waveguide Si–Ge avalanche photodiode," *Optica,* vol. 3, no. 8, pp. 793-798, 2016.

[20] M. Atef and H. Zimmerman, "Low-power 10 Gb/s inductorless inverter based common-drain active feedback transimpedance amplifier in 40 nm CMOS," *Analog Integrated Circuits and Signal Processing,* vol. 76, no. 3, p. 367–376, 2013.

[21] A. Mehrabian, Y. Al-Kabani, V. J. Sorger and T. El-Ghazawi, "PCNNA: A Photonic Convolutional Neural Network Accelerator," *arXiv:1807.08792 [cs.ET],* 2018.

[22] B. Sedighi, M. Khafaji and J. C. Scheytt, "Low-power 8-bit 5-GS/s digital-to-analog converter for multi-gigabit wireless transceivers," *International Journal of Microwave and Wireless Technologies,* vol. 4, no. 3, pp. 275-282, 2012.

[23] J. Fang et al, "A 5-GS/s 10-b 76-mW Time-Interleaved SAR ADC in 28 nm CMOS," *IEEE Transactions on Circuits and Systems,* vol. 64, no. 7, pp. 1673-1683, 2017.

[24] Micron Technology, Inc., "GDDR6 SGRAM MT61K256M32 8Gb: 2 Channels x16/x8 GDDR6 SGRAM," [Online]. Available: https://www.micron.com/products/graphics-memory/gddr6/part-catalog/mt61k256m32je-12. [Accessed 13 March 2019].

[25] J. Chiles, S. M. Buckley, S. W. Nam, R. P. Mirin and J. M. Shainline, "Design, fabrication, and metrology of 10 × 100 multi-planar integrated photonic routing manifolds for neural networks," *APL Photonics,* vol. 3, no. 10, p. 106101, 2018.

[26] SciPy, "scipy.signal.convolve2d," 10 February 2019. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.convolve2d.html. [Accessed 18 March 2019].

[27] Baidu Research, "DeepBench," [Online]. Available: https://github.com/baidu-research/DeepBench. [Accessed 16 March 2019].

[28] Advanced Micro Devices, Inc, "Radeon Vega Frontier Edition (Liquid-cooled)," [Online]. Available: https://www.amd.com/en/products/professional-graphics/radeon-vega-frontier-edition-liquid-cooled. [Accessed 16 March 2019].

[29] Advanced Micro Devices, Inc, "Radeon Instinct MI25 Accelerator," [Online]. Available: https://www.amd.com/en/products/professional-graphics/instinct-mi25. [Accessed 16 March 2019].

[30] NVIDIA Corporation, "NVIDIA Tesla P100 GPU Accelerator," [Online]. Available: https://images.nvidia.com/content/tesla/pdf/nvidia-tesla-p100-PCIe-datasheet.pdf. [Accessed 16 March 2019].

[31] NVIDIA Corporation, "Geforce GTX 1080 Ti," [Online]. Available: https://www.nvidia.com/en-us/geforce/products/10series/geforce-gtx-1080-ti/. [Accessed 16 March 2019].

[32] Vishay Intertechnology, "FETs As Voltage Controlled Resistors," 10 March 1997. [Online]. Available: https://www.vishay.com/docs/70598/70598.pdf.

[33] Z. You, J. Ye, K. Li, Z. Xu and P. Wang, "Adversarial Noise Layer: Regularize Neural Network By Adding Noise," *arXiv:1805.08000v2 [cs.CV],* 2018.

[34] L. Schares, C. Schubert, C. Schmidt, H. G. Weber, L. Occhi and G. Guekos, "Phase dynamics of semiconductor optical amplifiers at 10-40 GHz," *IEEE Journal of Quantum Electronics,* vol. 39, no. 11, 2003.

[35] M. J. Chen, C. S. Tsai and M. K. Wu, "Optical Gain and Co-Stimulated Emissions of Photons and Phonons in Indirect Bandgap Semiconductors," *Japanese Journal of Applied Physics,* vol. 45, no. 8B, p. 6576–6588, 2006.

[36] R. A. Budd et al., "Semiconductor optical amplifier (SOA) packaging for scalable and gain-integrated silicon photonic switching platforms," in *IEEE 65th Electronic Components and Technology Conference (ECTC)*, San Diego, 2015.

[37] A. Moscoso-Martir et al., "Silicon Photonics Transmitter with SOA and Semiconductor Mode-Locked Laser," *Scientific Reports,* vol. 7, no. 1, p. 13857, 2017.

[38] W3 Techs, "Usage of character encodings broken down by ranking," [Online]. Available: https://w3techs.com/technologies/character_encoding/ranking. [Accessed 23 November 2018].

[39] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2008,* pp. 1-70, 2008.

[40] Intel Corporation, "Intel 64 and IA-32 Architecture Software Developer Manual," 2018.

[41] ARM Limited, "ARM C and C++ Libraries Floating-Point Support User Guide," 2010.

Appendix A.1 Benchmarking Calculations for DEAP and DEAP-GIP

Table 9 contains the benchmarking parameters used in section 7.2. The speeds of the various GPUs were directly taken from [27], while the speed of the convolution was estimated using equation (26). Similarly, Table 10 contains the benchmarking parameters used in section 7.3 while the speed of convolution was estimated using equation (27). In some of the benchmarks, the kernels edge lengths were not equal, hence the parameters "R_w" and "R_h" which correspond to the width and height of the kernels. For each of the selected benchmarks, the parameters $R_m^2 C_m \leq 1024$, meaning that the convolutional is compatible with the DEAP implementations described in sections 5.4 and 6.5.

Table 9 - Benchmarking parameters for DEAP

W	H	C	N	K	R_w	R_h	P	S
700	161	1	4	32	5	20	0	2
112	112	64	8	128	3	3	1	1
7	7	832	16	256	1	1	0	1

Table 10 - Benchmarking parameters for DEAP-GIP

W	H	C	N	K	R_w	R_h	P	S
480	48	1	16	16	3	3	1	1
700	161	1	16	64	5	5	1	2

Appendix A.2 Selected DEAP simulator code

The entire source code for the DEAP simulator can be found at:

https://github.com/Viraj3f/DEAP. The following code simulates a dot product using a PWB:

```
class PWB:
    """
    A simple, time-independent model of a pwb.
    """
    def __init__(self, phaseShifts, outputGain):
        self.phaseShifts = np.asarray(phaseShifts)
        self.outputGain = np.asarray(outputGain)
        self.inputSize = phaseShifts.size

        mrr = MRRTransferFunction()
        self._throughput = mrr.throughput(self.phaseShifts)
        self._dropput = mrr.dropput(self.phaseShifts)

    def _update(self, newPhaseShifts, newOutputGain):
        assert self.inputSize == newPhaseShifts.size
        self.phaseShifts = np.asarray(newPhaseShifts)
        self.outputGain = np.asarray(newOutputGain)

        mrr = MRRTransferFunction()
        self._throughput = mrr.throughput(self.phaseShifts)
        self._dropput = mrr.dropput(self.phaseShifts)

    def step(self, intensities):
        intensities = np.asarray(intensities)
        if intensities.size != self.inputSize:
            raise AssertionError(
                    "Number of inputs ({}) is not "
                    "equal to  number of weights ({})".format(
                        intensities.size, self.inputSize))

        summedThroughput = np.dot(intensities, self._throughput)
        summedDropput = np.dot(intensities, self._dropput)
        photodiodeVoltage = summedDropput - summedThroughput

        return self.outputGain * photodiodeVoltage
```

The DEAP-GIP simulation code can be found at:
https://github.com/Viraj3f/DEAP/blob/master/examples/DEAP-GIP.ipynb

Appendix B. A Note on Semiconductor Optical Amplifiers

Semiconductor optical amplifiers (SOA) allow for the amplification of optical signals at gigahertz frequencies [34] without the need to convert to the electronic domain. Therefore, it seems that using a SOA for a PWB would be a better solution than using a TIA. The issue is that the silicon is an indirect bandgap semiconductor where the optical gain is only greater than the free carrier absorption at temperatures below $-250.15\ °C$ [35]. The other option would be to use a direct bandgap semiconductor but integrating that with a silicon platform is

challenging because of lattice mismatch. Some progress has been made towards implementing SOA with silicon [36, 37], but these solutions are generally complicated and expensive, so they are not expressed here. That said, SOA could be a viable alternative to electronic amplification if the integration with silicon becomes more convenient in the future.

Appendix C. Statement of Work

I state that all the work related to this thesis was done by me after September 1st, 2018. The original goal of this thesis was to write a simulator for photonic neural networks capable of solving ordinary differential equations. All of the work completed in the fall term was related to that original goal. In the winter term, the goal of the thesis was changed to designing a photonic architecture that can perform convolutions. Therefore, all of the work described in this report was completed from January 2019 onwards.

Appendix D. Floating Point Arithmetic using IEEE 754

Note: The technical standard for this thesis was in the fall term when the original goal was to write a software simulation for photonic neural networks.

All modern-day computer systems are built on top of the binary numeral system. Unsigned integers are directly represented using their base-2 representation, signed integers use two's complement and characters commonly use UTF-8 encoding [38]. For real numbers, IEEE-754 [39] is almost always used, as the most common instruction set architectures, x86 [40] for personal computers and servers and ARM [41] for embedded systems, both implement it.

IEEE-754 specifies that real numbers can represented in base-2 scientific notation by using single or double precision floats. The single precision format uses a 32-bit word to

represent a number, where bit 31 corresponds to the sign, bits 30 to 23 correspond to the exponent, and bits 22 to 0 correspond to the mantissa. The double precision format uses a 64-bit word, where bit 63 corresponds to the sign, bits 62 to 52 correspond to the exponent, and bits 51 to 0 correspond to the mantissa. This allows mathematical operations like multiplication and exponentiation faster and simpler to implement on CPU hardware. The standard also supports special values like NaN and $\pm$ and defines how they operate with other floating-point values.

My thesis will require a large number of floating-point computations, so understanding how IEEE-754 works is important. For instance, division by zero results in NaN, and operations that include NaN will result in NaN. This means that all critical software components should handle NaN values in a safe matter. The trade-offs between single and double precision floating points need to be known: single precision floating points can only represent numbers from $\pm(2 - 2^{-123}) * 2^{127} \approx 10^{38}$, whereas double precision floating points can store values from $\pm(2 - 2^{-52}) * 2^{1023} \approx 10^{308}$. Understanding the limitations of floating points is also important. Some decimals that round in base-10 notation do not round in base-2. This results in rounding errors. A simple example is the number 0.1, whose floating-point representation is stored as 0.10000000000000001 when using single floating-point precision. Rounding errors can become larger as the number of floating operations increase. The degree to which rounding errors are handled are dependent on the amount of precision required by the application. If no rounding error can be tolerated, one solution would be to store decimals as strings of integers at the cost of higher computational complexity. In my thesis, such a high degree of precision is not needed. Floating point equality can be dealt with by checking if two values are equal to each other within some degree of error. Overall,

single precision floating points will suffice well, and if errors are found, double precision floating points can be used instead.

Understanding floating point calculations is also important since my thesis proposes an architecture that operates on analog signals. A major difference is that floating point errors are deterministic and reproducible, whereas analog errors are due to stochastic shot, thermal and Johnson-Nyquist noise. Therefore, any physical components that perform arithmetic on analog components need to use produce as little noise as possible. This implies means that analog arithmetic is better for stochastic systems such as a neural network, whereas that floating-point arithmetic is better for deterministic systems like finance or banking.

Word count: 515