# LightML: A Photonic Accelerator for Efficient General Purpose Machine Learning

ISCA 2025 Submission #569 – Confidential Draft – Do NOT Distribute!!

## Abstract

The rapid integration of AI technologies into everyday life across sectors such as healthcare, autonomous driving, and smart home applications requires extensive computational resources, straining server infrastructure and incurring significant costs.

We present LightML, the first system-level photonic crossbar design, optimized for high-performance machine learning applications. This work provides the first complete memory and buffer architecture carefully designed to support the high-speed photonic crossbar, achieving more than 80% utilization. LightML also introduces solutions for key ML functions, including large-scale matrix multiplication (MMM), element-wise operations, non-linear functions, and convolutional layers. Delivering 320 TOP/s at only 3 watts, LightML offers significant improvements in speed and power efficiency, making it ideal for both edge devices and dense data center workloads.

## 1 Introduction

The booming AI market is producing numerous technologies that are quickly adopted and deployed in people's daily lives [2, 11, 42, 44, 46, 55, 61]. The sizes of these CNN, DNN, and Transformer-based AI models are rapidly expanding, demanding immense computational resources and pushing the server infrastructure to its limits, resulting in significant financial costs [5, 17, 70, 81]. Training and inference tasks for large language models can easily demand teraflops of computational power per inference task or training epoch [28]. Consequently, this "computational obesity" often restricts their deployment to large servers and data centers and frequently impedes user-side training and inference.

Computing with light offers a compelling alternative to conventional computing because optical interconnects, such as fibers and waveguides, are not constrained by energy efficiency issues related to Joule heating, RF crosstalk, and capacitance [30, 48, 60]. This eliminates the energy-bandwidth tradeoff that restricts the data transfer rate of electrical interconnects and allows an exceptionally high bandwidth density [8]. Various architectures have demonstrated the feasibility of photonic computing [3, 16, 43, 63, 85].

While these properties benefit the efficiency of today's machine learning applications, previous work has often focused on isolated physical aspects, lacking complete architectural support that integrates all necessary components. In particular, a few significant challenges remain to be addressed: 1. Existing work lacks a thorough memory solution to fully saturate the high data demand. For applications with complex data structures, such as large-scale matrix convolution, naive memory fetching schemes limit the performance of the crossbar arrays. 2. Current optical crossbar designs support limited operations, whereas general-purpose ML applications require a broader range of functionalities, including non-linear functions, element-wise operations, and normalization functions. 3. Prior designs do not fully utilize analog circuit design, resulting in redundant operations for basic tasks.

In this paper, we introduce LightML, an electronic-photonic co-designed architecture that optimizes memory access and supports various ML-specific operations to address existing gaps. The contributions of this work include:

- Pioneering the first system-level photonic crossbar architecture with a novel memory and buffer design that sustains high-speed data throughput.
- Integrating a non-linear function unit by utilizing phase modulators and achieving non-linear functions via the Fourier Series.
- Efficiently implementing operations such as matrix transpose, batch normalization, and ReLU activation functions through simple and easy-to-deploy circuit-level designs.

The advantages of LightML include its computing speed and high energy efficiency, offering 320 TOP/s computing power with only 3 watts of power consumption. It achieves 13× higher power efficiency than the GPU if considering the HBM power as well, and up to 1.8× higher efficiency compared to the state-of-the-art NVM-based crossbars. Additionally, with the proposed optimization, LightML can rival the NVIDIA A100 for ML model inference, providing up to 4× shorter latency. Furthermore, LightML delivers 5-bit precision while maintaining less than 3% inference accuracy loss compared to models in Floating Point. Therefore, LightML is an excellent choice for both edge devices, where energy efficiency is crucial, and data centers, where it can handle dense computing tasks efficiently.

## 2 Background

### 2.1 Photonic MAC Operation

Signed multiplication and addition can be performed optically using a simple beam splitter and differential photodetection. For multiplication, we make use of homodyne detection, which interferes with two optical fields with each other to achieve photo-electric multiplication at the quantum limit [24, 65]. When two coherent optical signals are incident on a 3db coupler, which outputs two light beam with 50% power,(Figure 1a), the instantaneous differential output intensity will be:

$$I_+ - I_- = 2|xy| \sin(\Delta\phi)$$

where $x$ and $y$ are the electric field amplitudes and $\Delta\phi = \phi_x - \phi_y$ is the relative phase difference between the two optical signals. Differential detection removes the common mode intensities of the input signals ($|x|^2$ and $|y|^2$), leaving only the product of the two field amplitudes, $|xy|$. Because the interference also depends on the relative phase difference between the two input signals, it is possible to encode both positive and negative numbers (e.g., $-y = |y|e^{j\pi}$)

'ISCA 2025', June 21–25, 2025, Tokyo, Japan
2024. ACM ISBN 978-1-4503-XXXX-X/18/06
https://doi.org/XXXXXXX.XXXXXXX

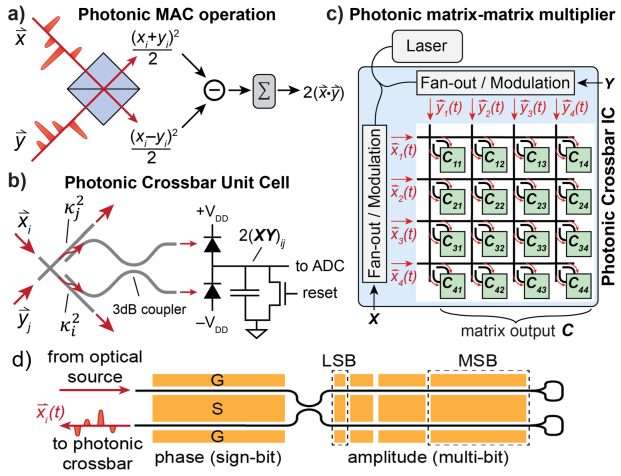


**Figure 1: a) Photonic MAC operations using a beam splitter and differential detection. b) Circuit-level implementation of a) using an on-chip beam splitter and integrating circuit. c) Photonic crossbar architecture for matrix-matrix multiplication comprised of an array of dot-product unit cells (green). d) Illustration of optical DAC comprised of a segmented Michelson modulator and phase shifter.**

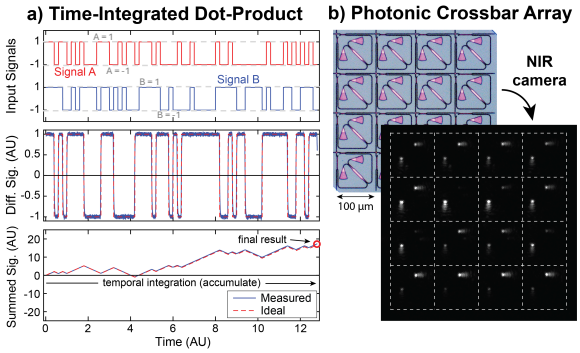


**Figure 2: a) Measured dot-product between two bipolar vectors (top). Difference signal measured by balanced photodetection (middle). Integrated difference signal gives the final result with a 3.6% error (bottom). b) Microscope image of $4 \times 4$ crossbar array (top) and captured NIR image of crossbar output (bottom).** Reproduced from [*#Hidden for blind review*].

and even complex numbers as recently demonstrated [35]. Since we encode data sequentially in the time domain, addition can be performed by temporal integration via charge accumulation on a capacitor, as shown in Figure 1b). Thus, the dot-product between vectors **x** and **y** can be accomplished optically on-chip with only a 3dB directional coupler and a few simple electrical components. This reduces both the input optical power and the ADC frequency by $1/N$, where $N$ is the number of time-multiplexed elements (i.e., length of the input vectors), thus significantly improving the scalability and compute efficiency over weight-stationary optical approaches [82].

## 2.2 Optical Modulator

The Michelson interferometric modulator (MIM) design [57] is comprised of a segmented intensity modulator with binary code weighting [77] (Figure 1d). By applying voltage to the segmented yellow cells, we can precisely control both the phase and amplitude of the modulated light. A $\pi$ phase shift ($y\, e^{j\pi} = -y$) is used to encode the sign of the input signal, while the amplitude is modulated linearly in proportion to the cell length. Different cell lengths are employed to represent the MSB and LSB. The use of segmented modulators allows for direct electro-optic (E-O) conversion of an $N$-bit binary value to an analog amplitude without the need for a dedicated DAC for each modulator. This can achieve high E-O conversion efficiency ($<$ 250 fJ/b) and better linearity than conventional optical modulators + electronic DAC designs [47, 77]. Additionally, the folded cavity design reduces footprint and capacitance by $\sim 2\times$ [57].

## 2.3 Photonic Crossbar

To scale this concept beyond single dot-product operations, we can create an array of dot-product unit cells, each coupled to row and column waveguides via directional couplers with a predetermined splitting ratio ($\kappa_i^2$ and $\kappa_j^2$) as illustrated in Figure 1c). This allows optical fan-out in a compact, two-dimensional array and amortizes the energy required for modulation. Similar arrays have been experimentally demonstrated with a large number of unit cells (e.g., $64 \times 64$ in [68] and $128 \times 128$ in [86]) in other contexts such as LIDAR [59, 86], photonic phased arrays [68], and in-memory photonic processors [16]. The crossbar can perform integer matrix-matrix multiplication (MMM) $\mathbf{C} = \mathbf{XY}$, where $X \in \mathbb{N}^{D \times P}$ and $Y \in \mathbb{N}^{P \times D}$, $P$ are the number of pulses temporally streamed into the array from memory, and $D$ can be the dimension of the crossbar array (up to 128). Previous work [82] shows that when the number of pulses, $P$, achieves 1024, the computational efficiency (TOPs/s/W) reaches the maximum. Our photonic crossbar architecture allows us to larger-scale MMMs with tiling, as we explore in Section 5.

The remarkable speed primarily stems from the ability to perform a dot product *at each crosspoint* of the photonic crossbar at speeds exceeding billions of multiply-accumulate (MAC) operations per second per crosspoint. In contrast, in conventional resistive memory-based crossbars, a dot product is executed in each *column* or *row* at speeds ranging from tens to a few hundred megahertz, limited by interconnect capacitance and analog noise.

# 3 Preliminary Result and Fabrication Feasibility

## 3.1 Preliminary Result

We have demonstrated the feasibility of our proposed architecture at both the unit cell and crossbar array levels [*#Hidden for blind review*]. A 4×4 homodyne multiplier device prototype was fabricated in the laboratory. The prototype utilizes non-resonant thermo-optic Mach-Zehnder interferometer (MZI) modulators with a tunable fiber laser source emitting 1550 nm light. Each cell in the array consists of two directional couplers to evenly distribute power from the row and column waveguides. A compact 3dB directional coupler connects to two output ports, which are further coupled to a near-infrared (NIR) image sensor via grating couplers. Results from a single unit cell are presented in Figure 2a. In this demonstration, two modulators were used to encode the phase of each signal, enabling the computation of the dot product between two 64-element bipolar

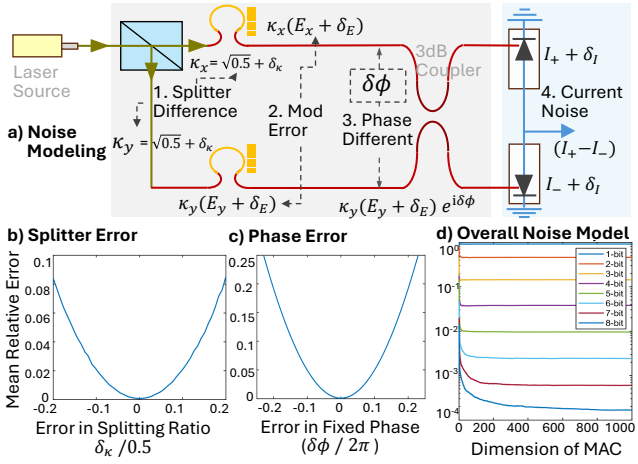


**Figure 3: a) Model of noise and fabrication error within each unit cell. b) and c) Ablation study of splitting ratio and phase noise. d) Overall Noise Modeling**

vectors. The differential signal from the two output grating couplers performs multiplication (middle panel), while temporal integration achieves accumulation (bottom panel). Additionally, we successfully demonstrated optical fan-out and NIR imaging of a passive $4 \times 4$ photonic crossbar array (Figure 2b). Post-processing the captured NIR image allows the extraction of MMM results, showcasing the scalability and practicality of our approach.

## 3.2 Error Modeling

The precision of the photonic crossbar is sensitive to noise and signal degradation. To assess the feasibility of industrial fabrication, we model the impact of noise from various fabrication non-idealities based on our photonic crossbar design. In Figure 3a, we present a preliminary error propagation analysis of optical MAC operations in a single unit cell. Assuming two digital input signals of dimension $N$, $\mathbf{x}, \mathbf{y} \in \mathbb{N}^N$, the ideal MAC output is $\mathbf{x}^T\mathbf{y}$. To evaluate the relative error, $\epsilon$, between the ideal output and the measured result from the ADC, we use the following equation:

$$\epsilon = \left(\mathbf{x}^T\mathbf{y} - (\mathbf{x}^T\mathbf{y})_{\text{measure}}\right) / \mathbf{x}^T\mathbf{y}$$

The measured output, $(\mathbf{x}^T\mathbf{y})_{\text{measure}}$, is the voltage measured from the accumulated circuit, which is $c_v \cdot \int \frac{1}{2}\left(I_+(t) - I_-(t)\right)\, \mathrm{d}t$. Here, the integral term $\int (I_+ - I_-)\, \mathrm{d}t$ represents the accumulated voltage over the MAC period, and $c_v$ is a calibration coefficient that scales the accumulated current to the ADC output. The noise modeling results shown in Figure 3 have been tailored for our architecture, drawing upon seminal theories of homodyne detection [53, 83] as well as recent developments in coherent photoelectric multiplication for AI processing [23].

**1. Beam Splitters** During optical fan-out, 50:50 beam splitters divide the laser into $2N$ beams with equal intensity before fanning-out these beams to each unit cell in a given row or column via directional couplers [82]. However, imperfections in splitting ratios during fan-out and distribution of the signals introduce power imbalances as the crossbar size increases. Each beam passes through $\log_2 2N$ 50:50 beam splitters and up to $N$ directional couplers, accumulating a fixed power imbalance due to deviation from the ideal splitting ratio. We model the power splitting ratio of each 50:50 splitter as $\kappa_x, \kappa_y = \sqrt{0.5} \pm \delta_\kappa$, while the coupling coefficients for each directional coupler in the crossbar can be written as $\kappa_n^2 = \kappa_{n+1}^2/(1+\kappa_{n+1}^2) + \delta_\kappa^2$ where $\delta_\kappa$ is the fixed fabrication error for each splitter/coupler sampled from a Gaussian distribution.

**2. Modulation Error:** Optical amplitude noise arising from analog electrical and optical noise in the modulators, driving circuitry, and the laser itself is present when encoding the optical inputs. The light intensity after modulation is modeled as $E_x + \delta_E$ and $E_y + \delta_E$, where $E_x \propto x$ and $E_y \propto y$ are the ideal electric field intensities, and $\delta_E$ is Gaussian noise.

**3. Phase Shift:** Variations in the optical path lengths between the two signals, $x$ and $y$, introduce fixed phase alignment errors. These errors occur due to minor differences in the physical paths traversed by the signals and can vary with temperature. Typically, the optical signals travel distances ranging from 1 to 10 mm between the modulator and the 3dB coupler. We model this phase noise as $e^{i\delta_\phi}$, where $\delta_\phi$ follows a Gaussian distribution, representing the random variations in phase alignment.

**4. Photon Detector:** Noise also occurs during optical-to-electrical conversion. The generated current is $I = \eta \cdot \frac{e}{h\upsilon}|E|^2$, where $\frac{e}{h\upsilon}|E|^2$ is the photon energy, and $\eta$ is the photon detector's conversion efficiency (typically between 0.5 and 0.8). We model photon detector noise as Gaussian noise $\delta_I$, affecting the output as $I_+ + \delta_I$ and $I_- + \delta_I$.

**5. Thermal Condition:** Photonic devices are sensitive to thermal variations because the refractive index of silicon changes slightly with temperature. In an RC circuit, thermal (Johnson–Nyquist) noise[39] induces voltage fluctuations across the capacitor, which can be expressed as: $V_{\text{rms}} = \sqrt{k_B T/C}$, where $k_B$ is the Boltzmann constant, $T$ is the absolute temperature, and $C$ is the capacitance. For instance, with $C = 15$fF (Same as DRAM) at room temperature (300 K), the voltage fluctuation is about 0.5 mV, which has only a minimal effect on our error model. Additionally, our platform minimizes thermal variations by consuming under 20W (including HBM2e), resulting in minimal temperature increases.

Other sources of noise, such as those from ADC readers, are negligible due to the maturity of these technologies. In Figures 3b) and 3c), we plot the relative dot-product error due to errors in the splitting ratios and phase error, respectively, while holding other noise sources constant. A smaller relative error ensures higher bit precision. For instance, a relative error below $1/2^4 = 0.0625$ allows for a 5-bit precise MAC operation (4 bits for magnitude and 1 bit for the sign). It is important to note that errors in both the splitting ratio and phase can be removed if: 1) they are constant after fabrication and 2) can be determined through an initial calibration step. In Figure 3d, we apply the Monte Carlo method to model the entire noise system. All noise sources follow Gaussian distributions with mean $\mu = 0$ and standard deviation $\sigma = 1/2^{N_b}$, where $N_b = \{1, 2, \ldots, 8\}$ represents different precision levels for devices (e.g., modulators, photon detectors). The relative error is plotted against the MAC operation dimensions, showing that the error decreases as the temporal dimension increases. Existing studies [62] have analyzed the impact of these physical effects in LightML on the

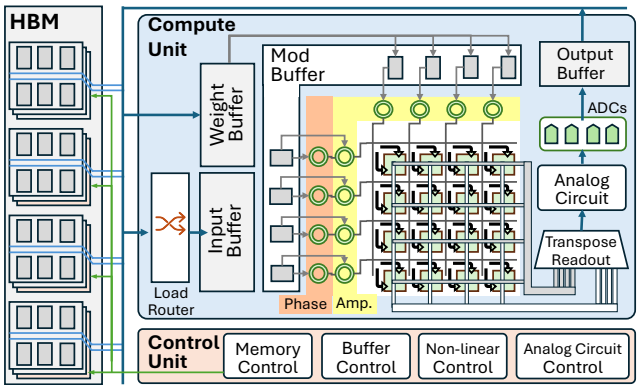


**Figure 4: Overview of LightML.**

accuracy of large-scale ML models. Simulation results in Table 6 show that the accuracy drop is around 1% ∼ 2%.

### 3.3 Industrial improvement

Our laboratory prototype is constrained by the fabrication technologies. However, industries with SOTA fabrication techniques can achieve higher precision and operating frequencies and address many limitations.

**Optical Modulator:** Existing work [78] has demonstrated a 20 GHz optical modulator fabricated using 40 nm technology, achieving both sign encoding and 32-level magnitude resolution. This corresponds to a 6-bit precision modulator.

**Phase Alignment:** For optical devices operating at a 1550 nm wavelength, the relative optical path length must be controlled to within $1550/(n_{eff} \cdot 2^4) \approx 50$ nm to maintain 4-bit amplitude and 1-bit sign. Achieving this requires laser trimming individual unit cells in the photonic crossbar array, as discussed extensively in [82]. Alternatively, localized thermal tuning has been demonstrated for post-fabrication trimming of silicon photonic devices [33]. Our lab is currently exploring using low-loss phase-change materials for laser trimming, building on methods shown in [75].

**Splitting Ratio:** To address optical losses and imperfections in splitting or coupling ratios and achieve 5-bit precision, the splitting ratio should maintain $\delta_\kappa^2 < 0.5/2^5$. A one-time calibration of the crossbar array can be performed. This calibration applies a scaling factor to each unit cell, compensating for fabrication variations and enabling the scalability of larger crossbar arrays.

Using SOTA technology, the precision of non-ideal components can exceed 5 bits, ensuring $\sigma < 1/2^5$. In Figure 3d), the relative error under 5-bit precision approaches $10^{-2} \approx 2^{-6}$, supporting results with approximately 6 bits of magnitude and 1 bit for the sign. Our design employs a conservative strategy, targeting 4 bits for magnitude and 1 bit for the sign, with potential scalability to 8 bits in future iterations. For instance, [85] experimentally demonstrated 8-bit precision across 1,000 optical MAC operations using high-performance foundry-fabricated modulators.

## 4 Overview

### 4.1 LightML Architecture

In Figure 4, we plot the architectural overview of LightML. The proposed architecture of LightML comprises of the core compute unit, the optical crossbar, supporting analog circuitry, memories and buffers, and control units, as depicted in Figure 4. The compute unit handles MMM, MVM, nonlinear functions, and element-wise operations. The memories and buffers are designed to support the high input/output data rates required by the compute unit, while the control unit is essential for orchestrating complex functions composed of multiple primitive operations, as listed in Table 1.

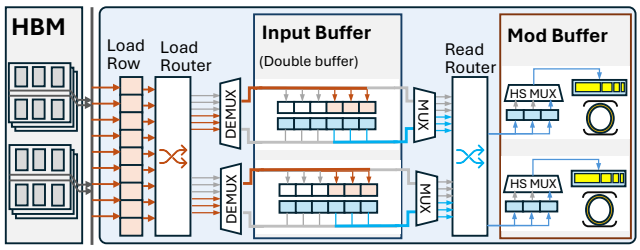


**Figure 5: Design Scheme of On-chip Input Buffers.**

**Computing Unit:** The crossbar is equipped with 2×128 modulators for amplitude modulation (in both dimensions) and 1×128 phase modulation (in the horizontal dimension), with the latter primarily used for computing non-linear functions. Each crosspoint includes an accumulation capacitor connected to analog-to-digital converters (ADCs) for processing readouts, as shown in Figure 1 (b). Column and row selectors are employed to reduce the number of ADCs by time-multiplexing them, allowing us to explore trade-offs between area, power, and latency. The ADCs and supporting circuitry will be further detailed to handle frequent operations such as divisions and transpositions (See Section 5.4).

**Control Unit:** The control unit consists of several components: a memory controller for address translation, a buffer controller to manage load/store routers and pipelining buffers, a nonlinear controller equipped with internal registers for storing coefficients and parameters required for non-linear functions, and an analog controller responsible for overseeing the circuits around the ADCs to perform regular, transpose, and scaling operations.

**Memory and On-chip Buffer:** The memories and buffers designed to meet the high bandwidth requirements of the compute unit, for a scalable crossbar, such as a 128×128 array with 2×128 modulators operating at 12 GHz, requiring $2 \cdot 128 \cdot 12$G= 3TB data per second to fully utilize the crossbar at the peak performance.

In Figure 5, we illustrate the data flow from memory to the modulator. LightML adopts a 2-stack High-Bandwidth Memory (HBM2e) configuration, providing up to 920 GB/s of memory bandwidth. For each memory cycle, HBM2e fetches 1KB of data for the chip. We implement a 1KB SRAM load row to store this memory on the chip.

We implement three on-chip buffers: the input buffer, the weight buffer, and the output buffer. Since the input buffers also supply data to the optical modulators, we will implement a double-buffer scheme. This allows one buffer to receive data from the HBM while the other feeds the modulator. In the next iteration, the two buffers will swap roles. This ensures that the HBMs can supply data to the input buffer within the duration of a single dot-product cycle. In Figure 5, memory data is written to the top blocks while the bottom blocks feed data to the modulators. These two buffers switch periodically, using DEMUXs to select which buffer to write to and MUXs to select which buffer to read from. The input buffer has

**Table 1: AI functions supported by LightML**

| Class | Function Name |
|---|---|
| Element-Wise Ops | Add, Sub, Mul, Scaling |
| Matrix/Vector Ops | Sum, Mean, Dot, MVM, MMM |
| Nonlinear Ops | Sin, Log, Exp, Tanh, Sigmoid |
| Normalization Ops | BatchNorm, LayerNorm |
| Systolic Ops | Linear, Conv2d |

128 lines serving 128 modulators, each containing 1KB of SRAM, totaling 128 KB.

Data from the loaded row (1KB) is distributively written into all 128 lines of the input buffer, with each line sequentially receiving 8B data. We implement a load router to control the data writing addresses, ensuring consistent addressing logic across all 128 lines. The weight buffer operates similarly to the input buffer but uses a single buffer architecture. When fetching the weights, the modulator operation is halted until the process is complete.

## 4.2 Flexibility in General AI Functions Support

AI models typically consist of complex modules—including Multi-Layer Perceptrons, convolutional layers, attention mechanisms, and various nonlinear activations.However, many of these modules can be decomposed into a small set of fundamental operations, such as MMM, MVM, element-wise operations, and matrix transpositions. For instance, the Transformer's attention mechanism involves MMM, softmax (a nonlinear function), element-wise scaling, addition, and matrix transposition [73]. The supported functions are summarized in Table 1. A majority of those primitive components can be efficiently implemented on our LightML, while the remaining operations are primarily memory-intensive. Implementing these operations enables accelerations for a wide range of AI models.

# 5 Common Modules for ML Applications

## 5.1 Quantization

To address the precision constraint, this work leverages the standard integer quantization machinist [18, 76]. The integer quantization utilizes batch-wise scaling factor, $s$, to map the real value $x \in [-S,\ S]$ into its 5-bit integer format, $\hat{x} \in [-16, 16]$:

$$\texttt{quant:}\ \hat{x} = \lfloor x \rceil / s\ , \quad \texttt{dequant:}\ x = s \cdot \hat{x}$$

The scaling factor $s$ is computed by $s = S/16$, where $S$ is the real-valued range of the input, and 16 is the integer range limit. During the MAC operation, all elements within each input matrix/vector $\hat{\mathbf{x}}$ and $\hat{\mathbf{w}}$ share the same scaling factors, $s_x$ and $s_w$. The output $\hat{y}$ is

$$\hat{y} = \sum_i^N \hat{w}_i\ \hat{x}_i = \sum_i^N s_w\ w_i\ \cdot s_x\ x_i = s_w\ s_x \cdot y$$

Hence, the integer output $\hat{y}$ has the combined scaling factor $s_w\, s_x$. In the worst-case scenario, $\hat{y}$ may require $2 \times 5 + \log_2(N) - 1$ bits to maintain accurate. In conventional digital computing, quantized outputs are typically stored in higher-precision formats, a new scaling factor is derived, and the data are then down scaled back to 5 bits, which is deployed in the cuBLAS library [51].

In LightML, down-scaling is efficiently handled by our ADC scaling unit (see Section 5.4). First, we conduct a one-time calibration of the laser power so that the maximum MAC output $S_{\max}$, obtained when 1,024 inputs are each set to the maximum value (+16), corresponds to 1V in the ADC readout. For a sequence of outputs with an individual scale, $S'$, we will adjust the ADC's reference voltage to $S'/S_{\text{MAX}}$V, fitting them to the suitable range.

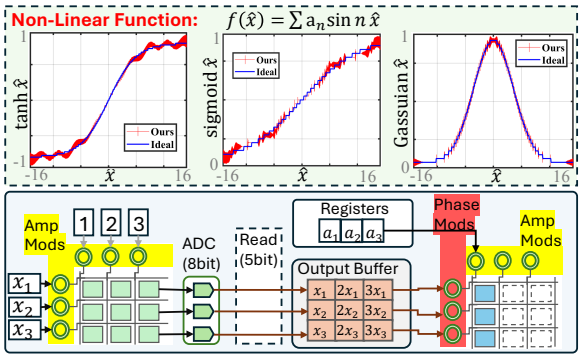


**Figure 6: Implementation for Nonlinear Functions.**

## 5.2 Non-Linear Function

Nonlinear functions, typically serving as activation functions in neural networks, vary by model. For instance, Long-Short-Term Memory (LSTM) [27] and Recurrent Neural Networks (RNN) [58] use *tanh*, while transformers [73] use the *Sigmoid* function (via SoftMax). Traditionally, they are implemented using math libraries with parallelism support, when available, on CPUs and GPUs. In ML accelerators, these functions are typically executed on dedicated hardware, which lacks the flexibility to easily adapt to new functions and algorithms. Previous work [67] uses a set of diodes to simulate nonlinear functions, but this approach can only implement a single nonlinear function and requires a carefully customized sequence of diodes, leading to increased latency.

In our approach, we implement nonlinear functions via the Fourier Series, and we use the phase modulator to generate sin or cos function outputs. The phase modulator calibrates the waveguide to ensure the phase difference between two light beams is 0 or $\pi$. To compute the nonlinear function, we couple two light beams: one containing only phase information and the other containing only amplitude information. The phase input, $\phi$, is represented as 5-bit data, dividing the phase into 32 intervals between $-\pi$ and $\pi$. The amplitude input is encoded as 5-bit data, $a$. After coupling, the current readout will be proportional to $a\sin(\phi)$. For any nonlinear function, $f(x)$, we can decompose it as a Fourier Series:

$$f(x) = \sum_{k=1}^{N} a_k \cdot \sin(2\pi kx/L) + b_k \cdot \cos(2\pi kx/L)$$

where the input, $x$, is within the interval $[-L, L]$, and $N$ is the degree of the series (number of terms). $a_k$ and $b_k$, are invariant to the input, $x$, and can be computed by:

$$a_k = \int_{-L}^{L} f(t) \sin\left(2\pi \frac{kt}{L}\right) \mathrm{d}t, \quad b_k = \int_{-L}^{L} f(t) \cos\left(2\pi \frac{kt}{L}\right) \mathrm{d}t$$

These coefficients are unique for each nonlinear function. For an odd function $f(x)$, the symmetric integral yields $b_k = 0$, and for an even function, $a_k = 0$. We preload the coefficients into the controller's register file. For a 5-bit precision digital input, the maximum degree for the Fourier Series is 32, and the register stores up to 64 5-bit coefficients for each nonlinear function.

Since the Fourier Series is periodic, it regulates the input within a specific interval. For each nonlinear function, we define a corresponding effective interval. The computation allows the input to be within this interval, $-L \le x \le L$ (e.g., using $L = 4$ for the

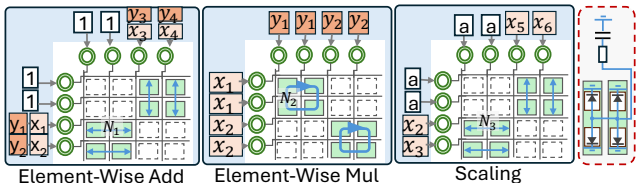


**Figure 7: Implementation for Element-Wise Operations.**

Sigmoid function), and all $x > L$ will be capped to $L$. The quantized input is a 5-bit data using $\hat{x} = \lfloor 16 \cdot \frac{x}{L} \rfloor$, and $-16 \leq \hat{x} \leq 16$. For all multipliers of $\hat{x}$, e.g., $2\hat{x}, 3\hat{x}, \cdots$, larger than 16, we modulate them by 32, which can be achieved by reading the last 5-bit of the 8-bit readouts.

In Figure 6, we illustrate the flow of nonlinear functions in three steps: 1. compute the multiplier: The input $x$ is modulated through an amplitude modulator and multiplied by the factors $1, 2, 3, \cdots, N$; 2. intermediate result measurement: The output is measured using 8-bit ADCs, with the last 5 bits extracted as the intermediate result and stored in the output buffer; 3. phase encoding and final multiplication: All $N$-degree of multipliers $\hat{x}, 2\hat{x}, \cdots N\hat{x}$ are encoded as phase information and multiplied by the corresponding coefficients $a_1, a_2, \cdots, a_N$. LightML is capable of batch computing up to 128 inputs simultaneously, significantly enhancing throughput for nonlinear operations. In the top of Figure 6, we use the 5-bit precise error model in Section3.2 to simulate three non-linear function: Sigmoid, tanh, and $e^{-x^2}$ (Gaussian). The blue curve represents the ideal function output in 5-bit precision, while the red error bar shows the error range (3-sigma) based on our implementation.

## 5.3 Element-Wise Operation

Element-wise operations, such as addition and multiplication, apply arithmetic operations to corresponding elements of two or more vectors or matrices. They are not only common but also essential in many AI models, particularly in skip connection, backpropagation, normalization, and attention mechanisms. The main challenge for LightML is the lack of data reuse, resulting in cell underutilization. Additionally, there is insufficient voltage accumulation in capacitors, causing ADC inaccuracies. Our design duplicates the element-wise operation across multiple unit cells and develops a new circuit to collect the current from all cells, enabling efficient element-wise operations on the crossbar.

**Addition:** On the left of Figure 7, we illustrate the addition of two vectors, $x$ and $y$. The inputs $x_i$ and $y_i$ are encoded as two consecutive pulses from a vertical or horizontal modulator, respectively. A modulator in the perpendicular direction generates constant light with unit amplitude, resulting in the output $1 \cdot x + 1 \cdot y$. To rapidly accumulate sufficient current, the unit light is duplicated $N_1$ times, ensuring that all $N_1$ cross-points (indicated by the colored blocks with blue arrows in Figure 7) produce the same output. This configuration allows for the computation of $2(128 - N_1)$ elements parallelly. Finally, we employ the circuit shown on the right of Figure 7, which redirects and accumulates the currents from all $N_1$ cells into a single capacitor, producing the final output.

**Multiplication:** In the middle of Figure 7, we show an example of element-wise multiplication between two vectors, $x$ and $y$. The vector $x$ is modulated from the horizontal direction, while $y$ is modulated from the vertical direction. For each input pair $x_i$ and $y_i$, $N_2$ modulators are used repeatedly to accumulate the current, and the final readout is the summation from $N_2^2$ unit cells.

**Scaling:** Scaling involves multiplying an input matrix or vector, $x$, by a constant factor, $a$. As shown on the right of Figure 7, this operation is similar to element-wise addition. We duplicate $\cdot N_3$ modulators from both directions to encode the coefficient $a$ and employ the rest $128 - N_3$ modulators to encode the input $x$. The results are obtained by summing the currents from $N_3$ unit cells.

The duplicate numbers $N_1, N_2$, and $N_3$ are determined using an optimal strategy to balance charge accumulation and parallelism. For a single cell, the charging duration is approximately 100 ns. For $N$ duplicated cells, this duration is linearly reduced to $\frac{100}{N}$ ns (In the real-world scenario, it might need $> \frac{100}{N}$ ns due to the imperfection of the circuit, requiring a further recalibration). As the duplicate number $N$ increases, parallelism is compromised. To maximize the number of element-wise operations within a fixed duration, e.g., 100 ns, the number of operations can be expressed as:

$$\text{Add:}(128 - N_1) \cdot N_1, \quad \text{Mul:}(128/N_2) \cdot N_2^2, \quad \text{Scal:}(128 - N_3) \cdot N_3$$

where $128 - N_1$ is the number of parallel inputs, and $N_1 = 100/(\frac{100}{N_1})$ is the number of operations for each cell that can be conducted in 100ns. We solve the optimal duplicate numbers to be $N_1, N_3 = 64$. This configuration enables 64 parallel inputs to be processed simultaneously, with a charging duration of 100/64 ns per cell. For multiplication, larger values of $N_2$ produce better results. To ensure consistency with addition and compatibility with the same accumulation circuit, we select $N_2^2 = 64$, resulting in $N_2 = 8$.

## 5.4 Analog-Implemented Functions

**ReLU Unit:** The Rectified Linear Unit (ReLU) is a widely used activation function in machine learning, defined as $\text{ReLU}(x) = \max(0, x)$, which introduces non-linearity into models. It can be efficiently implemented in an analog circuit, as a diode-based analog filter can nullify negative values. In LightML, the ReLU function is applied before the ADC readout. As the analog signal is transferred from the capacitor to the ADC, a rectifier diode blocks any negative current, ensuring that only positive values are passed.

**Transposable Readout:** Matrix transpose is frequently used in AI models such as transformers, e.g., BERT, GPT, natural language processing, and recommendation systems. Supporting efficient transposition is challenging for DRAM-based memory, as it requires $O(n^2)$ non-sequential read/write operations, disrupting data locality [79]. We propose performing transposition at the ADC readout circuit, offering a faster and more cost-effective solution compared to existing methods like the transpose unit [22] or in-plane transpose [21].

After the optical crossbar completes the MAC operation, the results are stored in capacitors as analog signals, and our approach transposes the intermediate output when needed for the next layer. We illustrate the readout process in Figure 8 a). Each of the 128×128 results is connected to a port that carries the analog voltage from the capacitors. Each crosspoint result is connected to row and column selectors. When transposition is needed, the selectors route the inputs: row selectors to column ADCs and column selectors to row ADCs, effectively performing the transposition. While an ideal setup would use one ADC per crosspoint, this approach can cause

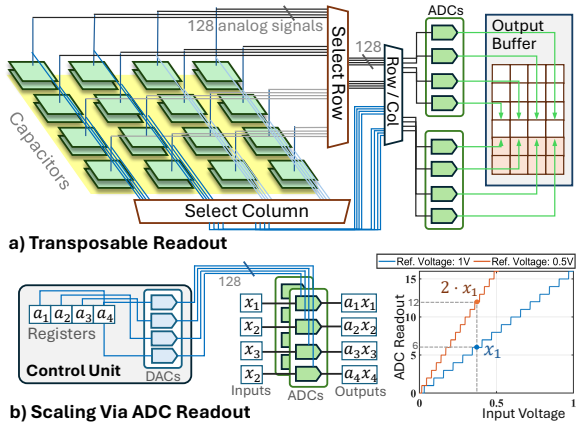


**Figure 8: a) Circuit Design for Transposable Readout Module. b) Implementation for Scaling Unit Via ADCs**

significant power consumption, especially with high-resolution and fast-sampling ADCs. LightML features 8×128 ADCs operating at 1GHz, and this process repeats for 16 rounds until all results are sent to the ADCs. The choice and number of ADCs are optimized to align with the throughput of the crossbar, output buffer, input buffer, and HBMs, details are in Section 7.4.

**Scaling via ADCs:** Section 5.3 introduces a general scaling method using the optical crossbar. We propose another cost-effective alternative using ADCs for scaling with a relatively small factor $a \in [0.25, 2]$, which is useful for frequently applied operations like batch normalization. An ADC converts continuous analog signals into discrete digital values, with the reference voltage determining the range of signals it can convert. This reference voltage, $V_{ref}$, typically represents the maximum voltage the ADC can measure and can be supplied externally or generated internally. The input voltage is mapped to a digital value based on its proportion to $V_{ref}$, following the formula: $\hat{x} = \frac{V_{in}}{V_{ref}} \times 2^n$ ($2^n$ is the maximum digital value for an n-bit ADC). Thus, the digital output is essentially $V_{in}$ divided by $V_{ref}$ and digitized. This relationship can be used to implement a division by a coefficient $a$ by setting $V_{ref} = a$, while the input is $V_{in} = x$. Note that $V_{ref}$ functions effectively within the range $a \in [0.25, 2]$, considering the error-tolerant range of an 8-bit ADC [52]. Figure 8b illustrates an example of scaling, along with the corresponding PSpice[49] simulation result. Here, the input $V_{in}$ ranges from 0 to 1 V, with $V_{ref} = 0.5$ V. The ADC output accurately reflects the input scaled by a factor of 2, demonstrating the precision of this approach.

## 5.5 Linear Transformation

A linear transformation is a fundamental operation in AI models, involving the multiplication of an input feature map with a weight matrix. Linear layers form the backbone of machine learning models, and when combined with batch normalization and ReLU activations, they become core building blocks of deep learning and neural networks. A general linear layer applies a transformation to the input data and typically includes MMM/MVM, Batch Normalization, and Activation Function, which can be expressed as:

$$y = \text{ReLU}\left(Wx/\sigma - \mu'\right)$$

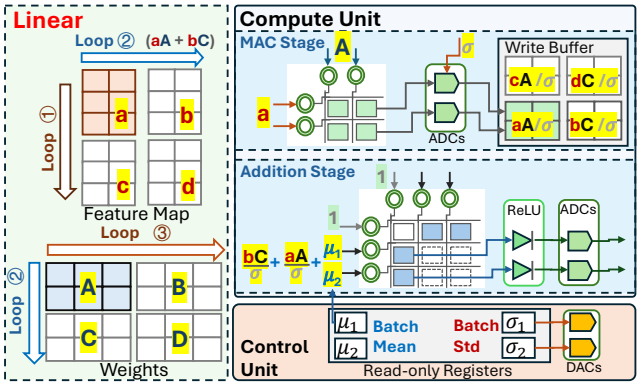


**Figure 9: Implementation of Linear Transformation.**

where $x \in \mathbb{R}^{C_{\text{in}} \times b}$ is the input matrix/vector, e.g. feature map, with dimension $C_{\text{in}}$ and $b$ is the number of batches, $W \in \mathbb{R}^{C_{\text{in}} \times C_{\text{out}}}$ is the weight matrix, and $\mu$ and $\sigma$ are the mean and standard deviation for batch normalization. Typically, the batch means and standard deviations are fixed during the inference [32] and are referred to as `running_mean` and `running_var` in Pytorch Platform [56].

**Matrix Tiling:** Building on existing MMM in photonic devices [23, 65, 82], LightML also introduces matrix tiling to handle large-scale MMM. The optical crossbar supports MMMs by, at maximum, two $128 \times 1024$ matrices (See Section 2.3), and tiling is necessary to accommodate MMMs with larger scales. Since LightML can perform MMM for two $128 \times 1024$ matrices at one time, MMM for larger matrices would need tiling, as depicted in Figure 9. We tile the feature map into sub-matrices with 128 batches and 1024 in-features and the weights into sub-matrices with 1024 in-features and 128 out-features. In each iteration, we compute the multiplication between these two tiled sub-matrices. We iterate the computation in three loops: ① iterates over the batches. We sequentially select 128 batches from the feature map and multiply them with the same weight sub-matrix; ② iterates over the input features. The weight matrix slides 1,024 dimensions from left to right, while the feature map slides from top to bottom; ③ iterates over the output features.

**Dataflow:** In Figure 9, both the input feature map and weights are tiled into $128 \times 1024$ submatrices to fit into the crossbar. This process has two stages: In the MAC stage, the sub-matrices $a$, $c$, $b$, and $d$ undergo the MAC operation with the sub-matrices $A$ and $C$ in the optical crossbar. The intermediate results, $aA$, $bC$, $cA$, and $dC$ of $128 \times 128$ matrices, are stored in the output buffer. In the addition stage, we perform an element-wise addition of the intermediate matrices. This involves summing the intermediate results to obtain $aA + bC$ and $cA + dB$.

**ReLU and BatchNorm:** The Batch Normalization consists of two operations: dividing $\sigma$ and substrating $\mu$, which can be performed through ADCs scaling, and then the element-wise addition. $\sigma$ and $\mu$ are initially stored in the coefficient registers of the control unit. During the MAC stages, $\sigma$ is converted into voltages as the reference voltages $V_{ref}$ to read the MAC output. During the addition stage, we append the negative of the batch mean, $-\mu'_1, -\mu'_2, \cdots, -\mu'_{C_{\text{out}}}$, as an extra term in the element-wise addition. Last, we use the analog ReLU unit to filter out negative values before ADC Readout.

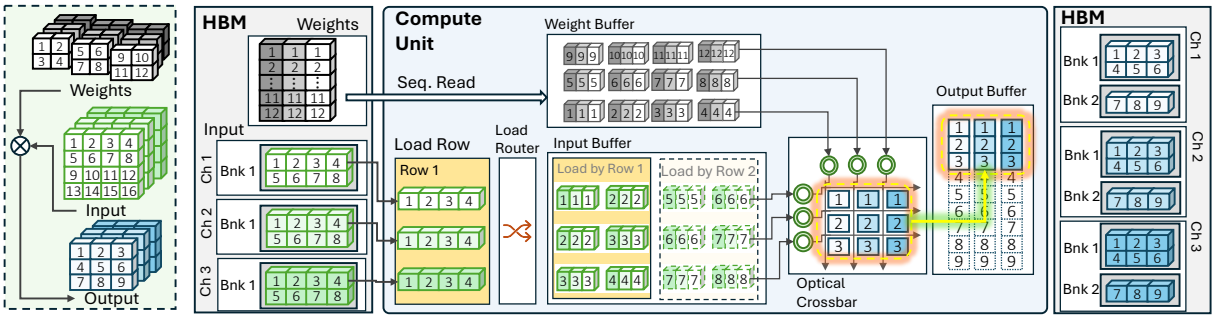


**Figure 10: Implementation of Convolutional Operation.**

## 5.6 Convolutional Operation

Convolution in machine learning is a critical component primarily for extracting features. The convolutional operation can be defined as $Y = W \circledast X$, where $X \in \mathbb{R}^{D \times D \times C_{\text{in}}}$ is the input feature map with width and height both $D$, and $C_{\text{in}}$ is the number of input channels. $W \in \mathbb{R}^{K \times K \times C_{\text{in}} \times C_{\text{out}}}$ is a four-dimensional tensor representing the weights, where $K$ is the kernel size and $C_{\text{out}}$ is the number of output channels. $Y \in \mathbb{R}^{D \times D \times C_{\text{out}}}$ is the output feature map.

**Optimized Memory Structure:** The proposed memory structure eliminates the cache-wise `im2col` operation. Instead, we utilize the transposable readout module, an on-chip address router, ensuring the computation unit and memory are efficiently utilized in most scenarios. A two-stack HBM2e contains 32 pseudo-channels and 8 banks for each, which can arbitrarily read 32KB data within each bank. We distribute the $D \times D \times C_{\text{in}}$ feature map across 32 pseudo-channels, with each channel storing $\lceil C_{\text{in}}/32 \rceil$ of $D \times D$ matrices.

Figure 10 shows an example of a $4 \times 4 \times 3$ feature map being convolved with $2 \times 2 \times 3 \times 3$ weights. We use digits to annotate the locations of elements in the feature map and different colors to represent different channels. In the example, the three-channel feature map is distributed across three pseudo-channels in the HBM, with their first bank, `Bnk1`, stores a $2 \times 4$ matrix, i.e., `1234-5678`. During memory loading, each channel loads a $1 \times 4$ matrix to the load row on the chip, forming a data line as `1234-1234-1234`. Next, we use the load router to reorder the data line into three lines: `111-222`, `222-333`, and `333-444` and write these lines into the input buffer. By repeating the memory loading process twice, we prepare the data for convolution. We then feed the unfolded feature map and weights (weights are preprocessed and sequentially read to the weights buffer) into the optical crossbar for the MAC operation. The MAC result is a $3 \times 3$ matrix. We use the transposable readout to read the mesh in a column-wise order, distributing results for different output channels into different buffer lines. The same data structure stores the results in the HBM, which will serve as the input feature map for the next layer.

## 6 Pipelining and Scheduling

To ensure all hardware units remain fully utilized, the computing speed of the crossbar must be aligned with the data transfer rates to and from the input and output buffers. A pipeline illustrating memory loads, stores, computation, analog circuit, and ADC readout is shown in Figure 11, where we capture a scheduling example of a MMM operation tiled into 4 submatrices. In the computing unit, the

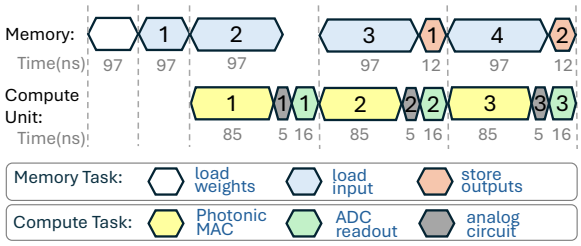


**Figure 11: Example of Scheduling for Linear Transformation**

crossbar's latency is proportional to the length of the dot-product operation, determined by the number of pulses modulated at each crosspoint. For a modulator operating at 12 GHz and a crossbar performing 1,024 continuous pulses, each dot-product operation takes approximately 85 ns. The analog circuit, including the ReLU unit, requires 5 ns to stabilize its output (Simulated Via PSpice). Additionally, the system incorporates 8×128 ADCs operating at 1 GHz, which need 16 rounds (16 ns) to complete the readout. On the memory side, feature maps are transmitted from HBM modules to the input buffers. A double-buffer scheme allows one buffer to receive data from the HBM while the other feeds the modulator. Since SRAM buffers are significantly faster than HBM, the bottleneck lies in the HBM read latency. For a sequential read of a $128 \times 1024$ matrix, the latency is approximately 97 ns. Subsequently, the results from the output buffer, a $128 \times 128$ matrix, are written back to the HBMs, requiring around 12 ns.

## 7 Evaluations

### 7.1 Experiment Setup

**Model and Dataset:** In this work, we evaluate the performance of LightML across multiple models and datasets. For the dataset, we use CIFAR-10 [37], CIFAR-100 [38], and ImageNet[12]. For the model, we use ResNet-18, ResNet-50, ResNet-101 [26], VGG-11, VGG-16, VGG-19 [64], MobileNet-V2, MobileNet-V3-Large [29]. We choose 32 as the batch size for the inference images, where the A100 GPU reaches its maximum efficiency.

**Simulation and Platform:** We implement LightML in a cycle-accurate simulator from scratch. We implement the modules for each proposed built-in function unit in the simulator. The simulator includes the memory module, the on-chip buffer, and the optical crossbar. The analog circuit, comprising the RC charging circuit

**Table 3: Hardware Configuration of LightML**

| Component | Size / # | Freq.(Hz) | Power | Area($mm^2$) |
|---|---|---|---|---|
| On-Chip Buffer | 256/128/64KB | 2G | 84mW | 0.68 |
| ADCs [52] | 128x8 | 1G | 2W | 1.5 |
| Capacitors | 128x128x2 | - | - | - |
| Laser Source[7] | 1 | - | 120mW | - |
| Modulators [78] | 128x2+128 | 12G | 810mW | 46.8 |
| Detectors[50] | 128x128x2 | 12G | 21.8mW | 235 |
| **On-Chip Total** | - | - | 3.03W | 284 |
| HBM [40] | 2x16GB | 1.6G | ~2x8W | ~2x80 |
| **Total** | - | - | ~19W | ~440 |

and ADC readout, is simulated using PSpice [49], while the power and area of the on-chip buffers are estimated with CACTI [4].

For the baseline comparison, latency results are measured on a real machine running the PyTorch platform [56]. The hardware setup includes an Intel i9-13900K for CPU, an NVIDIA A100 [9, 10] for GPU, and a Google TPU V3 [34].GPU and TPU measurements use `FP16` data precision, and CPU uses `Int8` (`Int8` and `Int4` are not widely adopted in CUDA or XLA environments, our methodologies are aligned with existing NVM-based crossbar work [25, 31, 66]).

## 7.2 Specification Detail

textbfOptical Layout: In our layout, we have assumed that each crossbar unit cell is 120$\mu$m x 120$\mu$m. This space includes the two directional couplers, 3 dB splitter, and on-chip balanced Ge photodetectors. A recent work by Rogers et al. [59] demonstrated a functional 512-pixel LiDAR receiver array on a silicon photonic platform that contained one directional coupler, 3 dB splitter, balanced Ge photodetectors, and optical antenna in an 80$\mu$m x 100$\mu$m unit cell. Notably, in this example, the optical antenna (excluded in our platform) comprised the majority of the layout space rather than the other optical components.

**Electrical Devices:** The RC charging circuit was designed using PSpice simulations to minimize charging duration while accumulating optical current to 1V with minimal noise. The circuit employs two optical-electrical transceivers [55], producing currents from -600 nA to 600 nA. It incorporates a 15 fF capacitor (similar to DDR4 technology), a 500 Ω resistor, and an 8-bit SAR-ADC operating at 1.2 GHz. After 1,024 pulse accumulations, the circuit's error distribution is characterized by $\mathcal{N}(0.00037, 0.0024)$.

Table 3 summarizes the hardware configuration of LightML. The on-chip buffer consists of three double buffers: an input buffer (256KB), a weight buffer (128KB), and an output buffer (64KB), operating at a frequency of 2GHz. LightML adopts 8 sets of 128 ADCs. Each crosspoint in the 128×128 array contains 2 capacitors, and its area is included in the Detector modules. The laser source[7] is integrated externally, operating at a wavelength of 1550nm, with adjustable power consumption up to 150mW. The modulators[78] are arranged in 128×2 for amplitude and 128 for phase. The modulator frequency can be scaled up to 50GHz, and we choose 12GHz to match the memory throughput. The homodyne detection scheme requires 2 photodetectors in every crosspoint of the 128×128 crossbar. Overall, LightML has a total power consumption of 3.04W and occupies an area of 284 $mm^2$. In addition to considering on-chip integrated HBM, we also investigate Micron's HBM2E [69]. The power consumption of HBM2E varies with temperature, supply voltage, and workload. For our analysis, we assume a supply voltage of 1V under full load at room temperature, which results in power consumption of 8W as reported in the technical documentation[40]. Moreover, based on data from the NVIDIA A100 [9], which utilizes an 8-stack HBM2E configuration—we estimate the package area to be approximately 80 $mm^2$.

## 7.3 Comparison with Existing Accelerators:

To the best of our knowledge, this is the first study to provide architectural support for photonic crossbars. Some previous studies have performed high-level evaluations but with limited architectural design (details in Section 9). In this work, we deliver a comparison with general-purpose ML processing via CPU, GPU, and TPU, and classical PE-based accelerator (ThinkFast[1]), PE-based PIM (SP-PIM [36]), and ReRAM-based crossbar RRAM-CIM[74] and RRAM-CIM2 [80]. The comparison includes performance efficiency (PE) and computational efficiency (CE).

Power consumption is a critical factor for scalability and deployment in edge devices. GPUs, TPUs, and ThinkFast consume 210W, 185W, and 300W, whereas LightML operates at only 3.04W. Note that, because the memory architectures differ across platforms, the reported power values exclude the memory subsystem. The performance efficiency (PE) and computational efficiency (CE) metrics highlight the benefits of LightML. With a PE of 105 TOP/s/W and a CE of 1.11 TOP/s/$mm^2$, LightML outperforms GPU by 84x and the state-of-the-art NVM-based accelerators by more than 1.84x in terms of efficiency. The comparison indicates that LightML offers a compelling combination of high performance, low power consumption, and efficient utilization of area, making it a suitable candidate for various ML applications, particularly in power-constrained environments.

**Table 2: Comparison of various AI accelerators across different Specs.**

| Platform | GPU | TPU | ThinkFast[1] | SP-PIM[36] | RRAM-CIM[74] | RRAM-CIM2[80] | LightML |
|---|---|---|---|---|---|---|---|
| Technology | 7nm | 12nm | 14nm | 28nm | 130nm | 40nm | 28nm |
| Frequency | 765MHz | 940MHz | 0.9GHz | 450MHz | - | 100MHz | 12GHz |
| Precision | FP16 | FP16 | Int8 | FP16 | Int2/4 | Int2/4 | Int5 |
| Peak Perf. | 312TFlOP/s | 180TFlOP/s | 820TOP/s | 3.2TOP/s | 1.8TOP/s | 0.63TOP/s | 320TOP/s |
| Area | 826$mm^2$ | 648$mm^2$ | 725$mm^2$ | 5.75$mm^2$ | 159$mm^2$ | 0.4$mm^2$ | 284$mm^2$ |
| Power | ~210W | ~185W | 300W | 0.14W | 45mW | 11mW | 3.04W |
| PE (TOP/s/W) | 1.48 | 0.97 | 2.73 | 22.4 | 40 | 57 | 105 |
| CE (TOP/s/$mm^2$) | 0.37 | 0.27 | 1.13 | 0.55 | 0.011 | 1.57 | 1.11 |

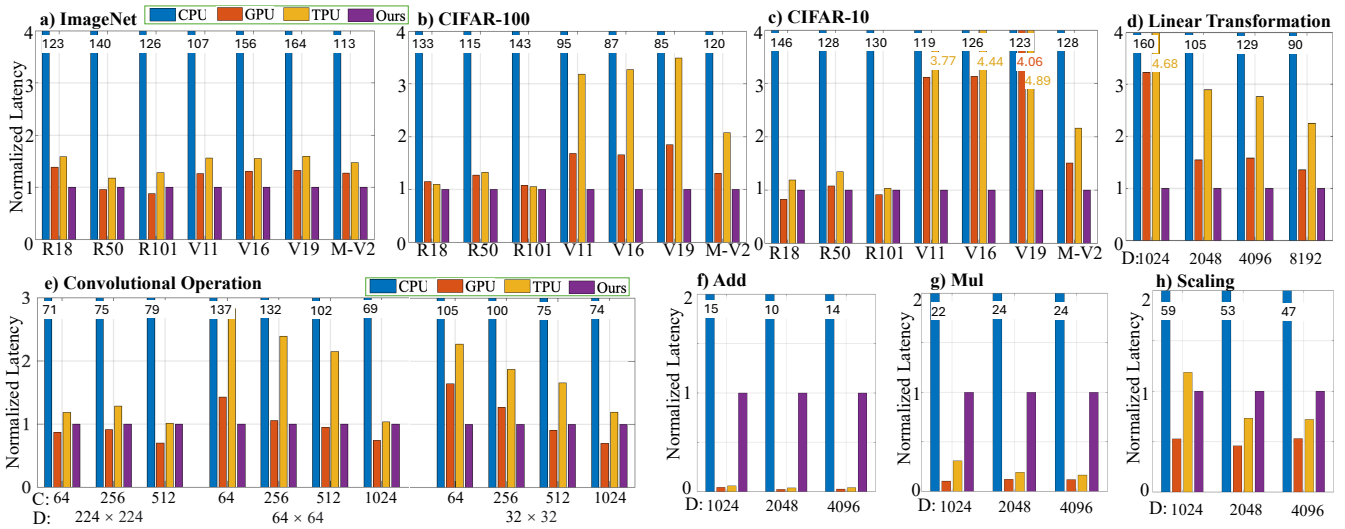


**Figure 12: (a), (b), and (c) show the normalized inference latency for ResNet18, ResNet50, ResNet101 (R18, R50, R101), VGG11, VGG16, VGG19 (V11, V16, V19), and MobileNet-V2 (M-V2) with 32-batch input images from ImageNet, CIFAR-100, and CIFAR-10; (e) present the latency for a $D \times D \times C \times 32$ feature map convolving with a $3 \times 3 \times C \times C$ kernel; (d), (f), (g), and (h) show the latency results for $D \times D \times 32$ inputs.**

## 7.4 Sensitive study – Number of ADCs

In this section, we explore the impact of varying the number of ADCs on power consumption and performance in LightML. As the readout period corresponds to the number of ADCs, the configuration of ADCs plays a crucial role in balancing power efficiency and performance. We compare five different configurations of ADCs: 1x128, 2x128, 4x128, 8x128, and 16x128. The results are summarized in Table 4. The power consumption increases with the number of ADCs, ranging from 1.27W to 5.03W. This increase is expected as more ADCs require more energy to operate. The peak performance also shows improvements as the number of ADCs increases from 140 TOP/s to 353 TOP/s. When evaluating performance efficiency (PE), we discover that the 4x128 configuration achieves the highest efficiency at 116 TOPS/W.

Additionally, we evaluate the performance of ML model inferences, specifically for ResNet18, 50, and 101 running on ImageNet with a batch size of 32. The latency and energy efficiency (measured as the number of images processed per joule) are detailed in Table 4. The results show that the 4x128 configuration not only provides the best PE but also demonstrates optimal efficiency for processing images. Overall, these findings highlight the importance of selecting an appropriate number of ADCs to optimize the balance between power consumption and performance in LightML.

**Table 4: Sensitive study of ADC Configuration**

| ADC Conf. | | 1x128 | 2x128 | 4x128 | 8x128 | 16x128 |
|---|---|---|---|---|---|---|
| Power (W) | | 1.27 | 1.54 | 2.03 | 3.03 | 5.03 |
| Perf. (TOP/s) | | 140 | 206 | 271 | 320 | 353 |
| PE (TOP/s/W) | | 110 | 133 | 133 | 105 | 70 |
| R18 | Latency | 9.4ms | 6.3ms | 4.8ms | 4.0ms | 3.7ms |
| | Image/J | 2196 | 2794 | 2864 | 2362 | 1614 |
| R50 | Latency | 39ms | 23ms | 16ms | 13ms | 11ms |
| | Image/J | 524 | 741 | 834 | 728 | 507 |
| R101 | Latency | 63ms | 39ms | 27ms | 23ms | 20ms |
| | Image/J | 325 | 449 | 493 | 413 | 283 |

## 7.5 Latency Comparison across ML Models/Ops

In this section, we compare the latency of LightML with that of CPU, GPU, and TPU. Latency data for CPU, GPU, and TPU is collected from real machines, the latency for the baseline machine is measured after the model and data are loaded to the device memory (`model.to(device)`), and we repeat the inference for 150 rounds and average the results for the last 100 rounds. GPU and TPU devices are synchronized before the start of each round. We use a batch size, `batch`, of 32 for all latency evaluations. We evaluate the normalized Latency by dividing the actual latency of each device by the latency of LightML.

Figures 12 f), 12 g), and 12 h) assess the normalized latency of Addition, Multiplication, and Scaling. The results show that LightML is not optimal for element-wise operations due to poor crossbar utilization, with at most 1/64 of the crossbar being used. Specifically, LightML is 10x slower than the NVIDIA A100 for Addition and Multiplication and 1.8x slower for Scaling.

Figures 12d) and 12e) evaluate the latency of the Linear and Convolutional (Conv) layers. The evaluation for convolution and linear layers integrates ReLU and Batch Normalization. The results show that LightML outperforms GPU by more than 1.4x for the linear layer, with a performance gain of up to 3.6x for smaller feature maps. For the convolutional layer, LightML slightly outperforms the GPU for $64 \times 64$ and $32 \times 32$ inputs, with performance gains ranging from 0.8x to 1.7x. However, for larger feature maps, our performance is inferior to that of the GPU.

Figures 12 a), b), and c) present the normalized latency results for ImageNet, CIFAR-100, and CIFAR-10, respectively. For ResNet models, the latency of LightML is approximately 0.85x to 1.2x compared to GPU. For VGG models, LightML outperforms the GPU with performance gains of 1.5x to 4.9x. The latency of the TPU

**Table 5: Comparison of Existing Non-Linear Function Units**

| NFU | Bits | Delay | Extra Power | Extra Area | Supported Functions |
|---|---|---|---|---|---|
| Zamanlogy [84] | 6 | 2.12ns | 64mW | 0.27mm$^2$ | $\sigma$ |
| PLAN[54] | 8 | 3.71ns | 57mW | - | $\sigma$ |
| RALUT [41] | 8 | 2.12ns | - | 1.06mm$^2$ | $\sigma$, tanh |
| Optim-LUT [45] | 8 | 2.82ns | 590mW | 0.78mm$^2$ | $\sigma$, tanh |
| PWL LUT [72] | 8 | 0.95ns | 466mW | 0.22mm$^2$ | $\sigma$, tanh |
| LightML | 5 | 3.1ns | N/A | 0 | Any |

is approximately 1.5x to 2x greater than that of the GPU due to differences in computing power, while the CPU is over 130x slower.

As shown in the results, LightML provides promising performance compared to GPU and TPU devices, operating at a power consumption of only 3W. However, there are two main drawbacks: 1. The crossbar dimension (128×128) is small compared to the GPU and TPU, where the A100 has 108×256 tensor cores, each delivering 4 FP16 operations per core [9, 10]. TPU v3 comprises four Matrix-Multiplication Units, each performing MMM for 128×128×2 input [34]. 2. LightML is not optimized for element-wise operations due to the lower data reuse rate. This issue also occurs in NVM-based crossbars. To address this problem, we can introduce an additional computing unit specifically for element-wise operations.

## 7.6 Comparison of Non-Linear Function Units

Prior non-linear function unit (NFU) in ML accelerators generally falls into two categories. The first category is modular approximation (e.g., piecewise linear (PWL) approximation) [54, 84], which segments a nonlinear function into several linear pieces. Although this method introduces only minor implementation overhead, it is limited to approximating a narrow set of nonlinear functions (typically only the sigmoid). The second approach employs Look-Up Tables(LUT) to pre-store a wider range of nonlinear functions, which incurs higher area and power costs. Other methods—such as classical 2nd or 3rd-order Taylor expansions—are often too costly to implement. LightML performs two rounds of ADC readouts for nonlinear function computation. In contrast, LightML performs nonlinear function computation using two rounds of ADC readouts. Similar to the element-wise operation, the NFU in LightML supports (12820)2 inputs concurrently—108 vertical modulators generate the phase and 20 horizontal modulators generate the amplitude, with a similar configuration in the other direction. No additional hardware is needed, as our optical modulators inherently produce sine and cosine functions, enabling the implementation of arbitrary nonlinear functions.

In Table 5, we compare the NFU of LightML against prior methods. Since the NFU in LightML can process 216 inputs per round, we use the same number of NFU units in the baseline for a fair comparison. Compared to existing approaches, LightML offers two key advantages: 1. It leverages existing hardware without requiring additional circuits, resulting in negligible power and area overhead. This simplifies fabrication and eliminates the need for data movement across different function units. 2. The optical NFU can implement a broad range of nonlinear functions using the same hardware, which is hard to achieve in prior methods.

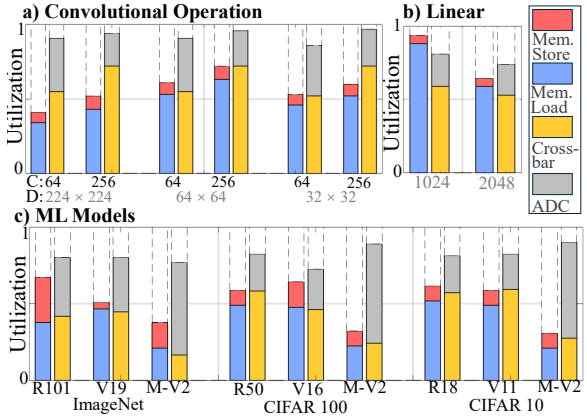


**Figure 13: Utilization of Memory / Computing Unit.**

## 7.7 Utilization Analysis of LightML

In this section, we analyze the utilization of three key components of LightML: memory load/store, optical MAC, and ADC readout. As discussed in Section 6, the utilization encompasses two parallel processes: (1) involves the memory process and (2) involves the computing unit.

The utilization results are depicted in Figure 13, and the notations and configuration are the same as Figure 12. The color bar shows the breakdown latency of each operation. For example, when we measure the individual latency of each process, and the utilization is calculated by dividing the component's latency by the overall latency. For memory operations, we display only the load and store latency, excluding the latency due to page faults and row activation, which are included in the overall latency but not shown in the figure. Figure 13 a) illustrates the utilization when computing the convolutional layer. For the convolution layer, the memory process shows under-saturation, with utilization rates between 40% and 60%. Conversely, the computing unit achieves more than 90% utilization, which demonstrates that the proposed convolutional module, utilizing the on-chip buffer, effectively manages complex memory access patterns, enabling the computing unit to operate at high utilization.

## 7.8 Impact of Precision on Model Accuracy

In this section, we evaluate the accuracy of ML models using quantization sampled from crossbar devices. We simulate $N_b = 5$ and $N_b = 8$ modes by quantizing the inputs and outputs of each MAC operation to 5 and 8 bits, respectively, and applying the same Gaussian noise as described in Section 3.2. In 5-bit mode, the error rate is low (0.0034), while in 8-bit mode the error rate reaches 0.632; however, the mean error distance is lower in the 8-bit mode. For simulating the NVM-based crossbar, we use 4-bit precision to encode the intermediate map and ($N_b = 2$ or $N_b = 3$)-bits for the weights. We assume separate functional units for addition and non-linear functions, providing 8-bit precision. All noise and precision loss in the simulation are generated using Gaussian noise.

The results, presented in Table 6, show the accuracy of different models under varying precision levels. The simulations were conducted on the PyTorch platform using a three-layer convolutional network for MNIST [13], ResNet-18 for CIFAR-10, and MobileNetV2

**Table 6: Accuracy under Different Precision Models**

| Dataset | MNIST | CIFAR-10 | ImageNet |
|---|---|---|---|
| NVM ($N_b = 2$) | 98.1% | 84.8% | 60.3% |
| NVM ($N_b = 3$) | 99.0% | 86.4% | 63.5% |
| LightML($N_b = 5$) | 99.2% | 90.6% | 66.1% |
| LightML($N_b = 8$) | 99.1% | 91.1% | 66.5% |
| GPU/TPU (FP16) | 99.2% | 92.4% | 69.8% |

for ImageNet. For MNIST, LightML matches the GPU/TPU performance, reaching 99.2% accuracy. For CIFAR-10 and ImageNet, LightML is slightly lower than the accuracy of GPUs/TPUs but significantly higher than the NVM-based crossbars.

## 8 for Large Language Models

Large Language Models (LLMs) [6, 14, 73] are widely used in machine learning but demand substantial computation and energy for training and inference. Recent studies show they tolerate lower precision, with only a 1–2% accuracy drop using FP8 and FP4 [20], making them viable for LightML deployment.

An algorithm example of an attention block is shown below. This block utilizes two LightML functions: LightML.Linear, which performs the linear transformation to compute `Q`, `K`, and `V` matrices; and LightML.NonLinear, which computes the softmax via a nonlinear function. Additionally, memory operations, such as head splitting and token padding, are simulated using a basic memory read/write strategy.

**Algorithm 1** Implementation for Attention Block

```
▷ x = Input Token
xq = LightML.Linear(x, W_q)
xk = LightML.Linear(x, W_k, transpose=True)
xv = LightML.Linear(x, W_v)
queries[:] = Memory.Split_Head(xq, num_head)
keys[:] = Memory.Split_Head(xk, num_head)
for i in range(num_head) do
    scores[i] = LightML.Linear(queries[i], keys[i], scale = √head_dim)
end for
scores = LightML.NonLinear(scores, 'softmax')
scores = Memory.Combine_head(scores, num_head)
output = LightML.Linear(scores, values)
```

We present a preliminary implementation of BERT[14], Llama 3.1 8B[71], and ViT/B-16[15], normalized to the performance of LightML as shown in Figure 14. We compare against two baselines—Google TPU and NVIDIA A100 GPU—both tested on real machines, with the GPU using `INT8` quantization. For BERT and Llama, tokenization is assumed to run on the CPU, processing input sequences of class digits. Sequences start with 8 tokens, with model completion up to 128 tokens; batch sizes are on x-axis. For the Vision Transformer, inputs are sampled from `ImageNet` with $384 \times 384$ pixel images and 1,000 classes.

Our current implementations lack proper optimizations for LLM models, resulting in performance disadvantages compared to the two baselines. In the Llama model, an NVIDIA A100 GPU is about 2.2× faster than LightML, but LightML achieves 6× higher energy efficiency per token. A detailed analysis of the attention block reveals two key inefficiencies. First, attention inputs, represented by the $Q$ and $K$ matrices of size $N_{\text{tokens}} \times D$, often have fewer token counts ($N_{\text{tokens}}$) than the crossbar dimension (128) for short sequences,

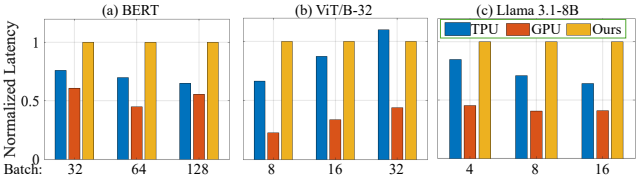


**Figure 14: LLM Inference Latency.**

leading to underutilization. Second, element-wise operations (e.g., addition) on large matrices are computationally inefficient on the crossbar, contributing 20% to attention overhead, whereas their impact on GPUs is negligible. Overall, while our results highlight the potential of photonic circuits for LLMs, further optimizations are needed and will be the focus of future work.

## 9 Related Work

Hamerly et al. [23] introduce an optical architecture that leverages homodyne detection to perform the MAC operation. In this scheme, the optical device relies on an electric controller, e.g., CPU, to prepare the input signal and process the nonlinear functions. They also show that both fully connected and convolutional neural network layers can be implemented by reformulating convolutions as matrix-matrix multiplications. The system consumes about 100aJ per MAC (50 mW) on a $100 \times 100$ crossbar array, excluding the power for memory, ADCs, and the laser source..

Sludds et al. [65] introduce Netcast, an optical architecture designed for edge computing. In this approach, model weights are encoded onto a multi-wavelength light beam using wavelength-division multiplexing (WDM). The cloud server transmits the encoded light to the edge device, where it is directed to an integrated crossbar array for processing. Netcast is demonstrated on a two-layer MLP operating on the MNIST dataset, achieving an optical energy consumption as low as 40aJ per operation (40mW).

Giamougiannis et al.[19] design a coherent photonic crossbar (Xbar) that maps each matrix element directly to a dedicated optical node. Instead of decomposing a matrix via SVD into cascaded unitary operations, the Xbar splits an input optical signal, modulates it with programmable amplitude and phase at each node, and then coherently recombines the signals. This mapping avoids exponential loss scaling, yielding linear loss dependence on individual node losses and enabling scalable linear transformations.

To the best of our knowledge, LightML is the first architectural design to support an optical computing device. It is meticulously designed to handle the high bandwidth of optical signals in and out of the crossbar, employs efficient pipelining for model execution, and is highly optimized to minimize computation and power consumption. Additionally, LightML is the first to implement general nonlinear functions within the optical crossbar—an achievement unmet by previous opto-electronic accelerators. This capability is especially relevant for today's rapidly evolving AI models.

## 10 Conclusion

We presented LightML, a high-performance photonic computing architecture optimized for ML applications. Supporting diverse ML operations, LightML is ideal for edge devices and data centers. It offers superior computing speed and energy efficiency, outperforming state-of-the-art solutions.

# References


[1] Dennis Abts, Jonathan Ross, Jonathan Sparling, Mark Wong-VanHaren, Max Baker, Tom Hawkins, Andrew Bell, John Thompson, Temesghen Kahsai, Garrin Kimmell, et al. Think fast: A tensor streaming processor (tsp) for accelerating deep learning workloads. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 145–158. IEEE, 2020.

[2] Shuroug A Alowais, Sahar S Alghamdi, Nada Alsuhebany, Tariq Alqahtani, Abdulrahman I Alshaya, Sumaya N Almohareb, Atheer Aldairem, Mohammed Alrashed, Khalid Bin Saleh, Hisham A Badreldin, et al. Revolutionizing healthcare: the role of artificial intelligence in clinical practice. *BMC Medical Education*, 23(1):689, 2023.

[3] Farshid Ashtiani, Alexander J. Geers, and Firooz Aflatouni. An on-chip photonic deep neural network for image classification. *Nature*, 606:501–506, 6 2022.

[4] Rajeev Balasubramonian, Andrew B Kahng, Naveen Muralimanohar, Ali Shafiee, and Vaishnav Srinivas. Cacti 7: New tools for interconnect exploration in innovative off-chip memories. *ACM Transactions on Architecture and Code Optimization (TACO)*, 14(2):1–25, 2017.

[5] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. Benchmarking tinyml systems: Challenges and direction. *arXiv preprint arXiv:2003.04821*, 2020.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[7] Bigeng Chen, Xingshi Yu, Xia Chen, Milan M Milosevic, David J Thomson, Ali Z Khokhar, Shinichi Saito, Otto L Muskens, and Graham T Reed. Real-time monitoring and gradient feedback enable accurate trimming of ion-implanted silicon photonic devices. *Optics express*, 26(19):24953–24963, 2018.

[8] Bill Corcoran, Mengxi Tan, Xingyuan Xu, Andreas Boes, Jiayang Wu, Thach G. Nguyen, Sai T. Chu, Brent E. Little, Roberto Morandotti, Arnan Mitchell, and David J. Moss. Ultra-dense optical data transmission over standard fibre with a single chip source. *Nature Communications*, 11:2568, 5 2020.

[9] NVIDIA Corporation. Nvidia a100 tensor core gpu architecture whitepaper, 2020. Accessed: 2024-06-11.

[10] NVIDIA Corporation. Nvidia a100 tensor core gpu datasheet, 2020. Accessed: 2024-06-11.

[11] Ivan Cvitić, Dragan Peraković, Marko Periša, and Brij Gupta. Ensemble machine learning approach for classification of iot devices in smart home. *International Journal of Machine Learning and Cybernetics*, 12(11):3179–3202, 2021.

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[13] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[16] Johannes Feldmann, Nathan Youngblood, Maxim Karpov, Helge Gehring, Xuan Li, M. Stappers, M. Le Gallo, Xin Fu, Anton Lukashchuk, Arslan S. Raja, Junqiu Liu, C. D. Wright, Abu Sebastian, Tobias J. Kippenberg, W. H. P. Pernice, and Harish Bhaskaran. Parallel convolutional processing using an integrated photonic tensor core. *Nature*, 589:52–58, 1 2021.

[17] Eva García-Martín, Crefeda Faviola Rodrigues, Graham Riley, and Håkan Grahn. Estimation of energy consumption in machine learning. *Journal of Parallel and Distributed Computing*, 134:75–88, 2019.

[18] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[19] George Giamougiannis, Apostolos Tsakyridis, Yangjin Ma, Angelina Totović, Miltiadis Moralis-Pegios, David Lazovsky, and Nikos Pleros. A coherent photonic crossbar for scalable universal linear optics. *Journal of Lightwave Technology*, 41(8):2425–2442, 2023.

[20] Zhuocheng Gong, Jiahao Liu, Jingang Wang, Xunliang Cai, Dongyan Zhao, and Rui Yan. What makes quantization for large language model hard? an empirical study from the lens of perturbation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 18082–18089, 2024.

[21] Fred G Gustavson and Tadeusz Swirszcz. In-place transposition of rectangular matrices. In *Applied Parallel Computing. State of the Art in Scientific Computing: 8th International Workshop, PARA 2006, Umeå, Sweden, June 18-21, 2006, Revised Selected Papers 8*, pages 560–569. Springer, 2007.

[22] Nastaran Hajinazar, Geraldo F Oliveira, Sven Gregorio, João Ferreira, Nika Mansouri Ghiasi, Minesh Patel, Mohammed Alser, Saugata Ghose, Juan Gómez Luna, and Onur Mutlu. Simdram: An end-to-end framework for bit-serial simd computing in dram. *arXiv preprint arXiv:2105.12839*, 2021.

[23] Ryan Hamerly, Liane Bernstein, Alexander Sludds, Marin Soljačić, and Dirk Englund. Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X*, 9(2):021032, 2019.

[24] Ryan Hamerly, Liane Bernstein, Alexander Sludds, Marin Soljačić, and Dirk Englund. Large-scale optical neural networks based on photoelectric multiplication. *Physical Review X*, 9:021032, 5 2019.

[25] Muhammad Abdullah Hanif, Aditya Manglik, and Muhammad Shafique. Resistive crossbar-aware neural network design and optimization. *IEEE Access*, 8:229066–229085, 2020.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] S Hochreiter. Long short-term memory. *Neural Computation MIT-Press*, 1997.

[28] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc.

[29] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[30] Junying Huang, Rongliang Fu, Xiaochun Ye, and Dongrui Fan. A survey on superconducting computing technology: circuits, architectures and design tools. *CCF Transactions on High Performance Computing*, 4(1):1–22, 2022.

[31] Ruirong Huang, Zichao Yue, Caroline Huang, Janarbek Matai, and Zhiru Zhang. Comprehensive benchmarking of binary neural networks on nvm crossbar architectures. *arXiv preprint arXiv:2308.06227*, 2023.

[32] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. pmlr, 2015.

[33] Hasitha Jayatilleka, Harel Frish, Ranjeet Kumar, John Heck, Chaoxuan Ma, Meer N Sakib, Duanni Huang, and Haisheng Rong. Post-fabrication trimming of silicon photonic ring resonators at wafer-scale. *Journal of Lightwave Technology*, 39(15):5083–5088, 2021.

[34] Norman P Jouppi, Doe Hyun Yoon, George Kurian, Sheng Li, Nishant Patil, James Laudon, Cliff Young, and David Patterson. A domain-specific supercomputer for training deep neural networks. *Communications of the ACM*, 63(7):67–78, 2020.

[35] Sadra Rahimi Kari, Nicholas A. Nobile, Dominique Pantin, Vivswan Shah, and Nathan Youngblood. Realization of an integrated coherent photonic platform for scalable matrix operations. *Optica*, 11:542, 4 2024.

[36] Jung-Hoon Kim, Jaehoon Heo, Wontak Han, Jaeuk Kim, and Joo-Young Kim. Sp-pim: A 22.41 tflops/w, 8.81 epochs/sec super-pipelined processing-in-memory accelerator with local error prediction for on-device learning. In *2023 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits)*, pages 1–2. IEEE, 2023.

[37] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[38] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-100 (canadian institute for advanced research).

[39] Rolf Landauer. Johnson-nyquist noise derived from quantum mechanical transmission. *Physica D: Nonlinear Phenomena*, 38(1-3):226–229, 1989.

[40] Seyed Saber Nabavi Larimi, Behzad Salami, Osman S Unsal, Adrián Cristal Kestelman, Hamid Sarbazi-Azad, and Onur Mutlu. Understanding power consumption and reliability of high-bandwidth memory with voltage underscaling. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 517–522. IEEE, 2021.

[41] Karl Leboeuf, Ashkan Hosseinzadeh Namin, Roberto Muscedere, Huapeng Wu, and Majid Ahmadi. High speed vlsi implementation of the hyperbolic tangent sigmoid function. In *2008 Third international conference on convergence and hybrid information technology*, volume 1, pages 1070–1073. IEEE, 2008.

[42] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[43] Xing Lin, Yair Rivenson, Nezih T. Yardimci, Muhammed Veli, Yi Luo, Mona Jarrahi, and Aydogan Ozcan. All-optical machine learning using diffractive deep neural networks. *Science*, 361:1004–1008, 9 2018.

[44] RT McAllister, Yarin Gal, Alex Kendall, Mark Van Der Wilk, Amar Shah, Roberto Cipolla, and Adrian Weller. Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning. International Joint Conferences on Artificial Intelligence, Inc., 2017.

[45] Pramod Kumar Meher. An optimized lookup-table for the evaluation of sigmoid function for artificial neural networks. In *2010 18th IEEE/IFIP International Conference on VLSI and System-on-Chip*, pages 91–95. IEEE, 2010.

[46] Sakorn Mekruksavanich and Anuchit Jitpattanakul. Lstm networks using smartphone data for sensor-based human activity recognition in smart homes. *Sensors*, 21(5):1636, 2021.

[47] Sajjad Moazeni, Sen Lin, Mark Wade, Luca Alloatti, Rajeev J. Ram, Milos Popovic, and Vladimir Stojanovic. A 40-gb/s pam-4 transmitter based on a ring-resonator optical dac in 45-nm soi cmos. *IEEE Journal of Solid-State Circuits*, 52:3503–3516, 12 2017.

[48] Sina Najmaei, Andreu L Glasmann, Marshall A Schroeder, Wendy L Sarney, Matthew L Chin, and Daniel M Potrepka. Advancements in materials, devices, and integration schemes for a new generation of neuromorphic computers. *Materials Today*, 59:80–106, 2022.

[49] James William Nilsson. *Introduction to PSpice® Manual, Electric Circuits: Using ORCad® Release 9.1.* Prentice Hall, Upper Saddle River, NJ, 2000.

[50] Kengo Nozaki, Shinji Matsuo, Takuro Fujii, Koji Takeda, Masaaki Ono, Abdul Shakoor, Eiichi Kuramochi, and Masaya Notomi. Photonic-crystal nanophotodetector with ultrasmall capacitance for on-chip light-to-voltage conversion without an amplifier. *Optica*, 3(5):483–492, May 2016.

[51] Cedric Nugteren. Clblast: A tuned opencl blas library. In *Proceedings of the International Workshop on OpenCL*, pages 1–10, 2018.

[52] Dong-Ryeol Oh, Kyoung-Jun Moon, Won-Mook Lim, Ye-Dam Kim, Eun-Ji An, and Seung-Tak Ryu. An 8-bit 1-gs/s asynchronous loop-unrolled sar-flash adc with complementary dynamic amplifiers in 28-nm cmos. *IEEE Journal of Solid-State Circuits*, 56(4):1216–1226, 2020.

[53] BM Oliver. Signal to noise ratios in photoelectric mixing. *Proc. IRE*, 49(12):1960, 1961.

[54] Zhe Pan, Zonghua Gu, Xiaohong Jiang, Guoquan Zhu, and De Ma. A modular approximation methodology for efficient fixed-point hardware implementation of the sigmoid function. *IEEE Transactions on Industrial Electronics*, 69(10):10694–10703, 2022.

[55] Trishan Panch, Peter Szolovits, and Rifat Atun. Artificial intelligence, machine learning and health systems. *Journal of global health*, 8(2), 2018.

[56] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[57] David Patel, Venkat Veerasubramanian, Samir Ghosh, Alireza Samani, Qiuhang Zhong, and David V. Plant. High-speed compact silicon photonic michelson interferometric modulator. *Optics Express*, 22:26788, 11 2014.

[58] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[59] Christopher Rogers, Alexander Y. Piggott, David J. Thomson, Robert F. Wiser, Ion E. Opris, Steven A. Fortune, Andrew J. Compston, Alexander Gondarenko, Fanfan Meng, Xia Chen, Graham T. Reed, and Remus Nicolaescu. A universal 3d imaging sensor on a silicon photonics platform. *Nature*, 590:256–261, 2 2021.

[60] R Saligram, A Raychowdhury, and Suman Datta. The future is frozen: cryogenic cmos for high-performance computing. *Chip*, 3(1):100082, 2024.

[61] Emre Sezgin. Artificial intelligence in healthcare: Complementing, not replacing, doctors and healthcare providers. *Digital health*, 9:20552076231186520, 2023.

[62] Vivswan Shah and Nathan Youngblood. Analogvnn: A fully modular framework for modeling and optimizing photonic neural networks. *APL Machine Learning*, 1, 6 2023.

[63] Yichen Shen, Nicholas C. Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, and Marin Soljačić. Deep learning with coherent nanophotonic circuits. *Nature Photonics*, 11:441–446, 6 2017.

[64] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[65] Alexander Sludds, Saumil Bandyopadhyay, Zaijun Chen, Zhizhen Zhong, Jared Cochrane, Liane Bernstein, Darius Bunandar, P Ben Dixon, Scott A Hamilton, Matthew Streshinsky, et al. Delocalized photonic deep learning on the internet's edge. *Science*, 378(6617):270–276, 2022.

[66] Shihao Song, Adarsha Balaji, Anup Das, and Nagarajan Kandasamy. Design-technology co-optimization for nvm-based neuromorphic processing elements. *ACM Transactions on Embedded Computing Systems*, 21(6):1–27, 2022.

[67] Nikita Stroev and Natalia G Berloff. Analog photonics computing for information processing, inference, and optimization. *Advanced Quantum Technologies*, 6(9):2300055, 2023.

[68] Jie Sun, Erman Timurdogan, Ami Yaacobi, Ehsan Shah Hosseini, and Michael R. Watts. Large-scale nanophotonic phased array. *Nature*, 493:195–199, 1 2013.

[69] Micron Technology. Hbm2e dram. https://www.micron.com/products/dram/hbm/hbm2e. Accessed: 2025-02-16.

[70] Neil C Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F Manso. The computational limits of deep learning. *arXiv preprint arXiv:2007.05558*, 2020.

[71] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[72] Ivan Tsmots, Oleksa Skorokhoda, and Vasyl Rabyk. Hardware implementation of sigmoid activation functions using fpga. In *2019 IEEE 15th International Conference on the Experience of Designing and Application of CAD Systems (CADSM)*, pages 34–38. IEEE, 2019.

[73] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[74] Weier Wan, Rajkumar Kubendran, Clemens Schaefer, Sukru Burc Eryilmaz, Wenqiang Zhang, Dabin Wu, Stephen Deiss, Priyanka Raina, He Qian, Bin Gao, et al. A compute-in-memory chip based on resistive random-access memory. *Nature*, 608(7923):504–512, 2022.

[75] Changming Wu, Haoqin Deng, Yi-Siou Huang, Heshan Yu, Ichiro Takeuchi, Carlos A Ríos Ocampo, and Mo Li. Freeform direct-write and rewritable photonic integrated circuits in phase-change thin films. *Science Advances*, 10(1):eadk1361, 2024.

[76] Hao Wu, Patrick Judd, Xiaojie Zhang, Mikhail Isaev, and Paulius Micikevicius. Integer quantization for deep learning inference: Principles and empirical evaluation. *arXiv preprint arXiv:2004.09602*, 2020.

[77] Xiaotie Wu, Bipin Dama, Prakash Gothoskar, Peter Metz, Kal Shastri, Sanjay Sunder, Jan Van der Spiegel, Yifan Wang, Mark Webster, and Will Wilson. A 20gb/s nrz/pam-4 1v transmitter in 40nm cmos driving a si-photonic modulator in 0.13µm cmos. pages 128–129. IEEE, 2 2013.

[78] Xiaotie Wu, Bipin Dama, Prakash Gothoskar, Peter Metz, Kal Shastri, Sanjay Sunder, Jan Van der Spiegel, Yifan Wang, Mark Webster, and Will Wilson. A 20gb/s nrz/pam-4 1v transmitter in 40nm cmos driving a si-photonic modulator in 0.13 $\mu$m cmos. In *2013 IEEE International Solid-State Circuits Conference Digest of Technical Papers*, pages 128–129. IEEE, 2013.

[79] Xin Xin, Yanan Guo, Youtao Zhang, and Jun Yang. Sam: Accelerating strided memory accesses. In *MICRO-54: 54th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '21, page 324–336, New York, NY, USA, 2021. Association for Computing Machinery.

[80] Jong-Hyeok Yoon, Muya Chang, Win-San Khwa, Yu-Der Chih, Meng-Fan Chang, and Arijit Raychowdhury. 29.1 a 40nm 64kb 56.67 tops/w read-disturb-tolerant compute-in-memory/digital rram macro with active-feedback-based read and in-situ write verification. In *2021 IEEE International Solid-State Circuits Conference (ISSCC)*, volume 64, pages 404–406. IEEE, 2021.

[81] Jie You, Jae-Won Chung, and Mosharaf Chowdhury. Zeus: Understanding and optimizing {GPU} energy consumption of {DNN} training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 119–139, 2023.

[82] Nathan Youngblood. Coherent photonic crossbar arrays for large-scale matrix-matrix multiplication. *IEEE Journal of Selected Topics in Quantum Electronics*, 29(2: Optical Computing):1–11, 2023.

[83] Horace P Yuen and Vincent WS Chan. Noise in homodyne and heterodyne detection. *Optics letters*, 8(3):177–179, 1983.

[84] Babak Zamanlooy and Mitra Mirhassani. An analog cvns-based sigmoid neuron for precise neurochips. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(3):894–906, 2016.

[85] H. Zhang, M. Gu, X. D. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. H. Yung, Y. Z. Shi, F. K. Muhammad, G. Q. Lo, X. S. Luo, B. Dong, D. L. Kwong, L. C. Kwek, and A. Q. Liu. An optical neural chip for implementing complex-valued neural network. *Nature Communications*, 12:457, 12 2021.

[86] Xiaosheng Zhang, Kyungmok Kwon, Johannes Henriksson, Jianheng Luo, and Ming C. Wu. A large-scale microelectromechanical-systems-based silicon photonics lidar. *Nature*, 603:253–258, 3 2022.