Dynamic Precision Analog Computing for Neural Networks

Sahaj Garg, Joe Lou, Anirudh Jain, Zhimu Guo, Bhavin J. Shastri, *Senior Member, IEEE*, and Mitchell Nahmias

(Invited Paper)

***Abstract*—Analog electronic and optical computing exhibit tremendous advantages over digital computing for accelerating deep learning when operations are executed at low precision. Although digital architectures support programmable precision to increase efficiency, analog computing architectures today only support a single, static precision. In this work, we characterize the relationship between the effective number of bits (ENOB) of precision of analog processors, which is limited by noise, and digital bit precision for quantized neural networks. We propose extending analog computing architectures to support dynamic levels of precision by repeating operations and averaging the result, decreasing the impact of noise. To utilize dynamic precision, we propose a method for learning the precision of each layer of a pre-trained model without retraining network weights. We evaluate this method on analog architectures subject to shot noise, thermal noise, and weight noise and find that employing dynamic precision reduces energy consumption by up to 89% for computer vision models such as Resnet50 and by 24% for natural language processing models such as BERT. In one example, we apply dynamic precision to a shot-noise limited homodyne optical neural network and simulate inference at an optical energy consumption of 2.7 aJ/MAC for Resnet50 and 1.6 aJ/MAC for BERT with $<2\%$ accuracy degradation, implying that the optical energy consumption is unlikely to be the dominant cost.**

***Index Terms*—Neural networks, signal-to-noise ratio, analog computing, optical computing.**

I. Introduction

ANALOG electronic and optical computing have demonstrated significant promise for accelerating matrix multiplications, the dominant computational cost in deep learning [1]. Common approaches include resistive crossbar arrays [2], [3], [4] and passive linear optical circuits, such as homodyne photoelectric multiplication and broadcast-and-weight [5], [6], [7], [8], [9]. By minimizing data movement costs (either using in-memory processing or moving data optically), amortizing energy over a large matrix multiplication, and completing entire matrix multiplications in a single clock cycle, analog processors can exhibit improvements over digital electronics in energy ($>10^2$), speed ($>10^3$), and compute density ($>10^2$) [1]. These performance improvements are critical as deep learning models double in size every 3.4 months [10], outpacing the growth of Moore's law and stretching the limits of digital hardware. However, analog computing exhibits these tremendous advantages over digital computing only when operations can be executed at low precision [1].

Fortunately, empirical research has demonstrated the robustness of deep learning at low bit precision. Despite being trained with 32 bit floating point representations, deep learning models can be deployed using just 4-8 bit integers without substantial accuracy degradation [11], [12], [13]. The minimum attainable precision is dependent on the network; larger neural networks, such as Resnet50, can be run at lower precision than compressed networks such as MobilenetV2 [12]. In addition, different layers of neural networks are tolerant to different levels of bit precision, and uniformly quantizing all layers to the same low precision leads to accuracy degradation. Using mixed bit precision, which executes precision-sensitive layers at high precision and insensitive layers at lower precision, enables inference with as few as 2-3 bits [13], [14], [15], [16], [17], [18], [19], [20], [21]. Digital hardware supports the execution of neural networks at dynamically varying bit precision; for example, NVIDIA A100 GPUs enable 1-64 bit integer arithmetic and 16-64 bit floating point arithmetic depending on the program or layer precision requirements [22].

By contrast, analog architectures for deep learning have not exploited dynamic precision to the extent that digital architectures have. Precision in analog computers is limited by various types of noise, such as shot noise, thermal noise, and weight read noise. We analyze the effective number of bits (ENOB) of precision of analog computing architectures in Section III. We find a surprising observation: because the dynamic range of outputs at each network layer varies, the effective number of bits of precision of an analog computing architecture is different for each layer. Unfortunately, precision-sensitive layers do not tend to have a higher ENOB by default. Despite this, analog matrix

Manuscript received 24 June 2022; revised 2 October 2022; accepted 24 October 2022. Date of publication 31 October 2022; date of current version 21 November 2022. The work of Zhimu Guo and Bhavin J. Shastri was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). *(Corresponding author: Zhimu Guo.)*

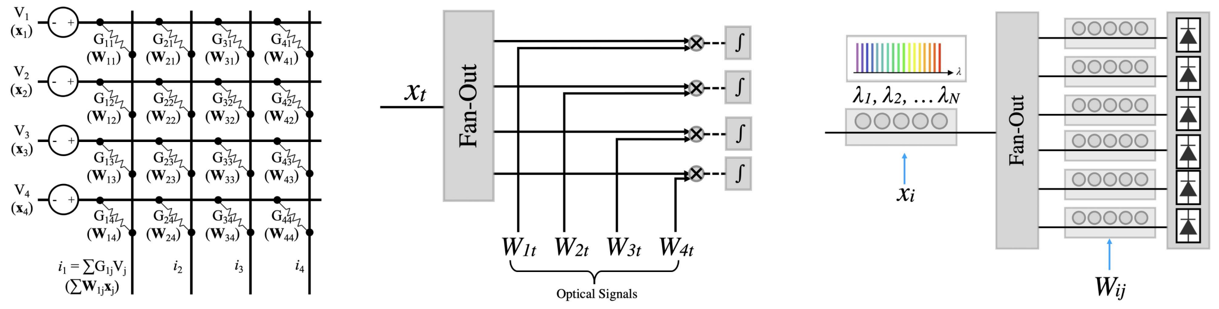
Sahaj Garg, Joe Lou, Anirudh Jain, and Mitchell Nahmias are with the Luminous Computing, Mountain View, CA 94305 USA (e-mail: sahajgarg@gmail.com; zlou@alumni.stanford.edu; anirudh1097@gmail.com; mitch@luminouscomputing.com).

Zhimu Guo is with the Department of Physics, Engineering Physics, and Astronomy, Queen's University, Kingston, ON K7L 3N6, Canada (e-mail: 15zg11@queensu.ca).

Bhavin J. Shastri is with the Department of Physics, Engineering Physics, and Astronomy, Queen's University, Kingston, ON K7L 3N6, Canada, and also with Vector Institute, Toronto, ON M5G 1M1, Canada (e-mail: shastri@ieee.org).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/JSTQE.2022.3218019.

Digital Object Identifier 10.1109/JSTQE.2022.3218019


Fan-Out
x_t
W_{1t} W_{2t} W_{3t} W_{4t}
Optical Signals
$\lambda_1, \lambda_2, \ldots \lambda_N$
x_i
W_{ij}
$i_1 = \sum G_{1j} V_j$ $(\sum W_{1j} x_j)$

(a) Resistive Crossbar Array (b) Homodyne Photoelectric Multiplication (c) Broadcast and Weight

Fig. 1. Three different approaches to analog electronic and optical computing that are used as case studies in this work. (a) Resistive crossbar arrays perform computing in-memory by applying inputs as voltages to rows of the array, and storing weights in flash, memristors, or phase change memory as the conductance between two points [2], [3], [4], [29], [30], [31], [32], [33], [34], [35]. Ohm's law yields a sum of products of these input voltages and weight conductances to produce a current that is proportional to the matrix-vector multiplication. (b) Homodyne photoelectric multiplication integrates charge at a coherent detector and accumulates MACs over time steps t [23]. (c) Broadcast and weight uses modulators and wavelength division multiplexing to compute a matrix-vector multiplication in a single clock cycle [36], [37], [38], [39], [40], [41], [42], [43], [44].

multipliers for deep learning do not dynamically vary precision to account for the precision sensitivity of different layers. At most, they allow for the amount of precision to be statically determined at design time for each neural network [1], [23], [24], or utilize mixed precision by using a digital processor for precision-sensitive operations, such as backpropagation or the first and last layer [3], [25], [26], [27], [28].

In this work, we propose extending analog computing architectures to support dynamic precision that can be selected by a programmer or compiler, analogous to bit precision in digital hardware. We observe that it is possible to trade off various performance metrics, such as energy efficiency, throughput, or area, to improve the precision of the analog computing engine. By repeating the same operation multiple times and averaging the results (as demonstrated by multi-memristive synapses [24]), precision can be improved at the cost of expending more energy. In Section IV, we discuss how redundant coding (repeating the same operation) in space or time can be applied to both analog electronic and optical computing architectures to enable dynamic precision.

A key challenge for deploying neural networks with dynamic precision is determining the optimal precision of different layers of the neural network given a hardware performance target. In Section V, we tackle this by solving an optimization problem. We focus specifically on the tradeoff between the energy per multiply-accumulate (MAC) of redundant coding and the resulting precision. We define a constrained optimization problem to maximize the original objective of the neural network subject to a constraint on total energy consumed, where the energy/MAC of each layer may be varied. The constraint is turned into a penalty function, and the optimal precision/energy tradeoff can be found by gradient descent.

We evaluate the advantages of supporting dynamic precision in analog computing on its energy consumption via software simulations in Section VI. We apply the method to a variety of convolutional neural networks (CNNs) for computer vision, including Resnet50, and natural language processing models such as BERT subject to different types of limiting noise, including shot noise, thermal noise, and weight noise, and determine the minimum energy/MAC with $<2\%$ accuracy degradation. First, we find that different networks are tolerant to different precision: some models require 34x more energy/MAC than others when each model is executed at uniform precision. This implies that fixed precision analog hardware will either be unable to support the networks that require greater energy/MAC, or will expend substantially more energy than needed for more noise-tolerant networks. Second, we find that using dynamic precision within a single network reduces energy consumption by up to 43-96% while obtaining similar accuracy. In one example, by using dynamic precision, optical homodyne photoelectric multipliers subject to the shot noise limit [23] can perform Resnet50 inference at 2.7 aJ/MAC and BERT inference at 1.6 aJ/MAC. This demonstrates that the energy floor for optical neural networks will be dominated by other costs, such as data movement, analog to digital conversion, and memory accesses.

II. Background and Related Work

A. Deep Neural Networks

A deep neural network consists of a series of layers, each of which performs a matrix multiplication followed by a nonlinear activation function. Let layer (l) have $N^{(l)}$ input neurons, $N^{(l+1)}$ output neurons, weight matrix $\mathbf{W}^{(l)}$, input vector $\mathbf{x}^{(l)}$, and nonlinear activation function f. Then the input to the $(l+1)^{th}$ layer is computed as:

$$\mathbf{x}_i^{(l+1)} = f\left(g\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}\right)\right) \quad 1 \leq i \leq N^{(l+1)}$$
$$g\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}\right) = \sum_{j=1}^{N^{(l)}} \mathbf{W}_{ij}^{(l)} \mathbf{x}_j^{(l)} \qquad (1)$$

Each neuron, as demonstrated above, computes a dot product between a row of the weight matrix $\mathbf{W}_i^{(l)}$ and the input to that layer, and performs $N^{(l)}$ multiply-accumulate operations (MACs). The neural network stacks L such layers and computes the probability of class labels as $p_m(y|\mathbf{x};\theta)$, where $\theta = \{\mathbf{W}^{(1)}, \ldots, \mathbf{W}^{(L)}\}$ denotes all the parameters of the neural network.

B. Low Precision Neural Networks

Neural networks are able to perform accurate inference at low bit precision in digital hardware. Empirical research has demonstrated that neural network accuracy degrades minimally when quantizing to 4 to 8-bit fixed point integer representations, despite the networks being trained using 32-bit floating point numbers [11], [12], [13].

A common method for quantizing floating point values to low precision is affine quantization [11]. In affine quantization, floating point inputs $\mathbf{x}^{(l)}$ (or weights) in the range $[\mathbf{x}_{\text{min}}^{(l)}, \mathbf{x}_{\text{max}}^{(l)}]$ are mapped to fixed point integers of B bits from 0 to $2^B - 1$ by scaling, translating, and rounding the inputs. Mathematically, this is

$$\begin{aligned} \mathbf{x}_q^{(l)} &= \text{round}\left(\frac{\mathbf{x}^{(l)}}{\Delta^{(l)}}\right) + z^{(l)} \\ \Delta^{(l)} &= \frac{\mathbf{x}_{\text{max}}^{(l)} - \mathbf{x}_{\text{min}}^{(l)}}{2^B} \\ z^{(l)} &= \text{round}\left(\frac{-\mathbf{x}_{\text{min}}^{(l)}}{\Delta^{(l)}}\right) \end{aligned} \tag{2}$$

The average precision required by commonly deployed neural networks can be lowered by using mixed precision. Different layers of neural networks are tolerant to different degrees of precision, and uniformly quantizing all layers of a neural network to the same bit precision leads to accuracy degradation [15]. There are many approaches for determining the bit precision of each layer, which requires searching an exponentially large space in the number of layers [13], [14], [15], [16], [17], [18], [19]. The method most similar to the one presented in this paper learns the bitwidth of each layer via gradient descent [21]. Other works perform post-training mixed precision quantization by making theoretical assumptions about signal-to-quantization noise ratio (SQNR) [20].

C. Noise in Analog Computing

A unique characteristic of analog computing is that it is subject to noise from many different sources. These noise sources include shot noise derived from Poisson distributed photons or electron fluctuations from the incoming signal [45], thermal noise in resistors, weight read noise in resistive memory from thermal noise, random telegraph noise, or $1/f$ noise [3], [46], and other types of device nonlinearities or fabrication variation [47]. If analog computing architectures quantize outputs to low bit precision, this noise may lead to bit errors in the least significant bit, which will not necessarily degrade neural network accuracy.

To account for nondeterminism in analog hardware, we replace the function g with a random variable $\tilde{g}$ that is sampled based on the noise distribution. The distribution of $\tilde{g}$ is dependent on the hardware architecture and type of noise. The noisy model's predictions are also a random variable, which we denote $\tilde{p}_m(y|\mathbf{x};\theta)$.

In this work, we model analog noise sources for three case studies: resistive crossbar arrays [2], [3], [4], [29], [30], [31], [32], [33], [34], [35], homodyne photoelectric multipliers [23], and optical broadcast and weight [36], [37], [38], [39], [40], [41], [42], [43], [44]. The details of these architectures are depicted in Fig. 1. In these case studies, a single matrix multiplication needs to be decomposed in smaller arrays of matrix multiplications, because the matrix multiplication unit has fixed size. The scaling of noise with the exact number of MACs is an approximation, since the noise will only increase in discrete units of the matrix multiplier size (i.e. 16-, 32-, 48-, etc element arrays).

Thermal Noise: First, we consider thermal noise that occurs from a transimpedance amplifier in receiver circuitry. We consider architectures that use digital inputs and outputs. Because the dot product is computed using quantized (i.e. normalized) versions of $\mathbf{W}$ and $\mathbf{x}$, the result $\mathbf{W}\mathbf{x}$ is recovered after rescaling the quantized product $\mathbf{W}^{(q)}\mathbf{x}^{(q)}$ by the range of $\mathbf{W}$ and $\mathbf{x}$. Thermal noise occurs as Gaussian noise with variance σ_t^2 (determined by receiver design) added to the quantized result, and is rescaled with the signal. Because longer dot products are computed via partial sums, noise variance grows linearly with the number of MACs $N^{(l)}$. So, we write:

$$\begin{aligned} \tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}\right) &\sim \sum_{j=1}^{N^{(l)}} \mathbf{W}_{ij}^{(l)} \mathbf{x}_j^{(l)} \\ &\quad + \xi\sqrt{N^{(l)}}\left(\mathbf{W}_{\text{max}}^{(l)} - \mathbf{W}_{\text{min}}^{(l)}\right)\left(\mathbf{x}_{\text{max}}^{(l)} - \mathbf{x}_{\text{min}}^{(l)}\right)\sigma_t \\ \xi &\sim \mathcal{N}(0,1) \end{aligned} \tag{3}$$

These equations for thermal noise apply to both resistive crossbar arrays and optical broadcast-and-weight since both encode the signal in the current at the receiver.

Weight Noise: For weight noise in resistive memory, we assume that the weights are quantized and write

$$\begin{aligned} \tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}\right) &\sim \sum_{j=1}^{N^{(l)}} \left(\mathbf{W}_{ij}^{(l)} + \xi_j\left(\mathbf{W}_{\text{max}}^{(l)} - \mathbf{W}_{\text{min}}^{(l)}\right)\sigma_w\right)\mathbf{x}_j^{(l)} \\ \xi_j &\sim \mathcal{N}(0,1) \quad 1 \le j \le N \end{aligned} \tag{4}$$

Shot Noise: The magnitude of shot noise, however, is signal dependent. For homodyne photoelectric multipliers using analog inputs and weights, we have from the derivation in [23]:

$$\begin{aligned} \tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}\right) &\sim \sum_{j=1}^{N^{(l)}} \mathbf{W}_{ij}^{(l)} \mathbf{x}_j^{(l)} + \xi \frac{\|\mathbf{W}_i^{(l)}\|_2 \|\mathbf{x}^{(l)}\|_2}{\sqrt{N^{(l)}}}\sigma_s \\ \xi &\sim \mathcal{N}(0,1) \end{aligned} \tag{5}$$

Many other systems and noise sources can be represented using this phenomenological framework. These include unitary optical matrix multiplication [7] and optical switching based architectures [48] in both the thermal and shot noise limited regimes, resistive crossbar arrays subject to nonlinear weight noise, and optical systems with Relative Intensity Noise (RIN) [1], among others. In many of these cases, the noise is not linear, such as when noise is added to power, but signals are encoded in amplitudes [7].

While prior works in analog computing discuss the varying noise tolerance of different computations and propose using mixed-precision, to our knowledge, this is the first work that extends analog computing to support programmatically dynamic precision. Prior work statically increases the precision for a single neural network by using greater energy/MAC [1], [23], [24], simulates hardware that injects noise into only a single layer of the neural network [23], or uses mixed analog and digital computing, where precision-sensitive operations such as the first and last layer [3], backpropagation [28], and others [25], [26], [27] are computed digitally at high precision while other operations are computed subject to analog noise. Other approaches include using arithmetic codes to correct errors in analog computing [49]. Recent work in stochastic computing has introduced dynamic precision to improve efficiency [50].

III. Relating Noise and Bit Precision

We characterize the effective number of bits (ENOB) of precision of analog computing architectures. Following the definition in [51], the ENOB is the number of bits of an ideal analog to digital converter (ADC) for which the RMS quantization error is equal to the RMS analog noise. In the case of quantizing neural networks to low bit precision, each layer is quantized to a different full scale range, so the ideal ADC quantizes over a different range per layer. If all layers are instead quantized to the same range, neural network inference accuracy substantially drops. As a result, the ENOB varies from layer to layer. To validate this relationship, we simulate the error rates of neural networks subject to analog noise or quantized to the equivalent ENOB.

For each layer, we measure the RMS of an ideal ADC. When using uniform quantization, all activations in layer (l) are quantized in the range $[\mathbf{x}_{\max}^{(l)}, \mathbf{x}_{\min}^{(l)}]$. When quantizing to B bits, the RMS of quantization error of the ideal ADC is $\frac{1}{\sqrt{12}}(\mathbf{x}_{\max}^{(l)} - \mathbf{x}_{\min}^{(l)})/2^B$.

The effective number of bits is derived by equating the RMS of analog noise and RMS of the ideal ADC. We get

$$\mathrm{ENOB}^{(l)} = \log_2\left(\frac{\mathbf{x}_{\max}^{(l)} - \mathbf{x}_{\min}^{(l)}}{\sqrt{12}\,\mathrm{RMS}_a^{(l)}}\right) \tag{6}$$

For example, the ENOB for for thermal noise using the variance in (3) is:

$$\mathrm{ENOB}^{(l)} = \log_2\left(\frac{\left(\mathbf{x}_{\max}^{(l)} - \mathbf{x}_{\min}^{(l)}\right)}{\sigma_t\left(\mathbf{W}_{\max}^{(l-1)} - \mathbf{W}_{\min}^{(l-1)}\right)\left(\mathbf{x}_{\max}^{(l-1)} - \mathbf{x}_{\min}^{(l-1)}\right)\sqrt{12\,N^{(l-1)}}}\right) \tag{7}$$

We evaluate whether ENOB accurately capture precision by testing whether they predict the inference accuracy of neural networks subject to analog noise. To do so, we evaluate each layer of Resnet50 subject to thermal noise with varying σ_t, and measure the ENOB in each layer. We then remove thermal noise, and instead run each layer at low bit precision using its respective ENOB.[1] We report results in Table I, including the average ENOB across all layers. We find that the accuracy subject to analog noise and quantization error closely match one another, despite the distributions of noise being different. The quality of the approximation decreases at extremely low precision, where uniform errors from a quantizer are no longer have similar effects as additive white noise.

TABLE I
Thermal Noise and Simulated ENOB for Resnet50

Noise ($1000\sigma_t$)	Noisy Accuracy	Average ENOB	Low-Bit Accuracy
7.1	36.5	3.2	39.1
4.5	66.4	3.8	61.7
3.2	71.2	4.3	70.0
2.2	73.2	4.8	73.1
1.8	73.8	5.1	73.9
1.6	74.1	5.3	74.2
1.4	74.4	5.4	74.5
1.0	74.9	5.9	75.0
0.7	75.3	6.4	75.2
0.4	75.4	7.1	75.4
0.0	75.5	8.0	75.5

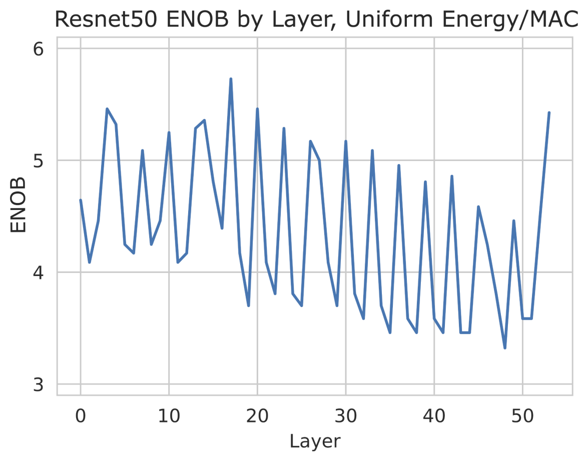
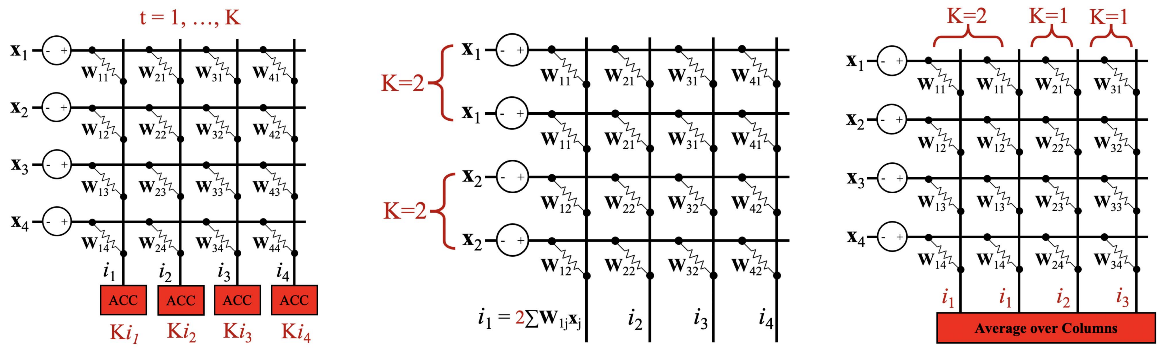


Fig. 2. Effective number of bits of precision when using fixed thermal noise σ_t for different layers of Resnet50, average ENOB 4.8.

We plot the ENOB for each layer of Resnet50 in Fig. 2. We note a surprising observation: even when using a fixed σ_t, the ENOB in different layers varies substantially. This is because the ENOB is a function of the dynamic range compression from inputs to outputs of layer (l), as explicitly described in (7).

While the ENOB varies for different layers, there is no guarantee that precision sensitive layers are executed at high precision and vice versa. A heuristic evaluation demonstrates this is not the case: the first and last layer typically require the highest precision [3], but are not allocated higher precision in Fig. 2. To address this problem, we propose a method to vary the precision settings of analog computers in Section IV and a method to determine the optimal precision for different layers in Section V.

[1]To more closely capture the ENOB, we allow for quantization to any discrete number of quantization bins, as opposed to bits. For example, quantization over 25 uniformly spaced bins requires 4.644 bits.


t = 1, …, K
x_1 x_2 x_3 x_4
W_{11} W_{21} W_{31} W_{41} W_{12} W_{22} W_{32} W_{42} W_{13} W_{23} W_{33} W_{43} W_{14} W_{24} W_{34} W_{44}
i_1 i_2 i_3 i_4
ACC ACC ACC ACC
Ki_1 Ki_2 Ki_3 Ki_4
K=2
K=2
$i_1 = 2\sum W_{1j}x_j$
K=2 K=1 K=1
Average over Columns

(a) Time Averaging (b) Spatial Averaging per Matrix (c) Spatial Averaging per Dot Product

Fig. 3. Dynamic precision with redundant coding; resistive crossbar arrays are used as an illustrative example. Changes to the architecture are shown in red. We use K to denote the number of times an operation is repeated, where in (a) operations are repeated for K clock cycles, in (b) the same inputs and weights are repeated, and in (c) only certain rows of $\mathbf{W}$ are repeated.

IV. Dynamic Precision With Redundant Coding

We propose extending analog computing architectures to support dynamic precision through a general method called redundant coding. Redundant coding entails performing the same computation multiple times, either in different spatial channels of the analog matrix-vector multiplier, or over multiple clock cycles, and averaging the result. This reduces the impact of noise on the computation at the expense of other performance metrics, such as energy/MAC, throughput, or compute density. Redundant coding has previously been demonstrated as a method for improving the precision of analog computation by using multiple memristors to encode the same weight [24]. This work generalizes redundant coding as a technique applicable to all analog electronic and optical computing architectures, and proposes designing architectures that can programmatically vary the amount of redundancy, as opposed to statically improving the precision with a fixed amount of redundancy.

We first demonstrate how redundant coding can be used to vary precision at the granularity of a matrix multiplication. To enable time averaging, receiver circuitry may add an accumulator, and the compiler can instruct the hardware to accumulate the same computation for K clock cycles and average the result before requantizing. To enable spatial averaging, K devices may be used to encode the same weights and inputs in a single dot product. For example, in resistive crossbar arrays, multiple resistive memory elements in a column can be used to encode the same weight, and the same input voltage can be broadcasted to multiple rows. The broadcasting of weights and inputs to multiple devices can be determined at compile time. The application of redundant coding to a resisitve crossbar array via time and spatial averaging are shown in Figs. 3(a) and 3(b). With K times redundant coding, both of these approaches effectively compute

$$K\mathbf{W}\mathbf{x} = \begin{bmatrix}\mathbf{W} & \mathbf{W} & \dots & \mathbf{W}\end{bmatrix}\begin{bmatrix}\mathbf{x}\\ \mathbf{x}\\ \vdots\\ \mathbf{x}\end{bmatrix}$$

$$\mathbf{W}\mathbf{x} = \begin{bmatrix} W_{11} & W_{12} & \dots & W_{1N}\\ W_{21} & W_{22} & \dots & W_{2N}\\ \vdots & & \ddots & \\ W_{N1} & W_{N2} & \dots & W_{NN}\end{bmatrix}\begin{bmatrix}x_1\\ x_2\\ \vdots\\ x_N\end{bmatrix}$$

Dynamic precision at the finer granularity of each row of the weight matrix requires modifying spatial averaging to use a varying number of dot product engines to repeat different computations. Then, the architecture will need to average over a programmable number of dot product engines. This can either be performed by configuring receiver circuitry to support an accumulator over variable numbers of engines, or by leveraging a digital arithmetic logic unit (ALU) and carrying operations at low bits within the ALU. The configuration of these accumulators or the instructions for averaging can be determined at compile time. We illustrate spatial averaging for dot products in a resisitve crossbar array in Fig. 3(c). Similar approaches have been used by resistive crossbar arrays to split the computation of different bits over different columns [4]; here, we propose that the accelerator do so dynamically.

The precision of the computation varies with the degree of redundancy. Because the amount of redundancy linearly increases the energy/MAC, we parameterize precision with respect to $E^{(l)}$, the amount of energy/MAC used for the $(l)^{th}$ layer of the neural network. We consider the ideal case where this quantity may be continuously modulated, as opposed to taking one of several quantized energy levels. In each of the cases discussed in Section II-C, the signals add linearly, but the noise sources add in quadrature, so the noise standard deviation is proportional to $\frac{1}{\sqrt{E^{(l)}}}$. We note that the type of averaging also determines other tradeoffs that are made; time averaging trades off throughput, and spatial averaging trades off area.

Thermal Noise: We replace (3) with:

$$\tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}, E^{(l)}\right) \sim \sum_{j=1}^{N^{(l)}} \mathbf{W}_{ij}^{(l)}\mathbf{x}_j^{(l)} + \xi\sqrt{N^{(l)}}\left(\mathbf{W}_{\max}^{(l)} - \mathbf{W}_{\min}^{(l)}\right)\left(\mathbf{x}_{\max}^{(l)} - \mathbf{x}_{\min}^{(l)}\right)\frac{\sigma_t}{\sqrt{E^{(l)}}} \quad (8)$$

Weight Noise: We replace (4) with:

$$\tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}, E^{(l)}\right) \sim \sum_{j=1}^{N^{(l)}} \left(\mathbf{W}_{ij}^{(l)} + \xi_j \left(\mathbf{W}_{\max}^{(l)} - \mathbf{W}_{\min}^{(l)}\right) \frac{\sigma_w}{\sqrt{E^{(l)}}}\right) \mathbf{x}_j^{(l)} \quad (9)$$

For thermal noise and weight noise, $E^{(l)}$ is equivalent to K in this case as a unitless constant because the free parameters σ_t and σ_w are determined by the engineering of a given architecture.

Shot Noise: For shot noise, we may modify (5) to have $E^{(l)}$ represent a physical, not relative, energy quantity for specific architectures such as [23]. In this case, $E^{(l)}$ is measured in Joules, $E^{(l)}\lambda/(hc)$ is the average number of photons per MAC, and the output subject to shot noise in homodyne photoelectric multipliers [23] is:

$$\tilde{g}\left(\mathbf{W}_i^{(l)}, \mathbf{x}^{(l)}, E^{(l)}\right) \sim \sum_{j=1}^{N^{(l)}} \mathbf{W}_{ij}^{(l)} \mathbf{x}_j^{(l)} + \xi \frac{\|\mathbf{W}_i^{(l)}\|_2 \|\mathbf{x}^{(l)}\|_2}{\sqrt{N^{(l)} E^{(l)} \lambda/(hc)}} \quad (10)$$

V. Learning Optimal Precision-Energy Tradeoffs

A key challenge for deploying neural networks with dynamic precision is determining the optimal precision of different layers of the neural network given a hardware performance target. In this work, we focus primarily on the tradeoff between energy/MAC and precision resulting from redundant coding. To do so, we propose to solve a constrained optimization problem to maximize the original objective of the neural network subject to an energy constraint. This technique is similar to prior work on learning the bit precision of different layers of mixed-precision neural networks in digital hardware [21]. Note that this optimization problem is solved for a pretrained network and only optimizes over the energy allocated to each layer, so it does not retrain the neural network. We use $E_{\max}$ to denote the energy budget, and $\mathbf{E}$ to denote the vector of all energies per MAC to be learned across layers $1, \ldots, L$, i.e. $(E^{(1)}, \ldots, E^{(L)})$. The total energy consumed by the network can be computed from the number of MACs in each layer, $n_{mac}^{(l)}$, as $E_{tot}(\mathbf{E}) = \sum_{l=1}^{L} E^{(l)} n_{mac}^{(l)}$. The objective of the neural network, in this case log likelihood, is evaluated on ordered pairs of inputs (i.e. images) and outputs (i.e. classification labels) $(\mathbf{x}, y)$ sampled from the data distribution $p_d(\mathbf{x}, y)$. Then, the optimization problem is

$$\begin{aligned} \underset{\mathbf{E}}{\text{minimize}} \quad & -\mathbb{E}_{(\mathbf{x},y)\sim p_d}\left[\log \tilde{p}_m(y|\mathbf{x}; \theta, \mathbf{E})\right] \\ \text{s.t.} \quad & \sum_{l=1}^{L} E^{(l)} n_{mac}^{(l)} \leq E_{\max} \end{aligned} \quad (11)$$

We apply several transfrmations to the optimization problem so that it may be solved by gradient descent. We address the fact that $\tilde{p}_m$ is a random variable by using the reparameterization trick [52]. We can treat the noise $\xi_i^{(l)} \sim \mathcal{N}(0,1)$ as inputs to the network, in which case $\tilde{p}_m$ becomes a deterministic function of the noise, weights, and inputs. We use ξ to denote the random vector of all noise sources. In addition, we turn the linear constraint into a penalty in the objective via the Lagrange multiplier penalty method [53]. Finally, we find that in practice, penalizing the logarithm of total energy consumption (an equivalent quantity) is more stable for optimization. This is because it is advantageous to have energy allocations and the log likelihood to be similar orders of magnitude in the loss, but the energy allocations change by orders of magnitude during training. This yields our final optimization problem,

$$\begin{aligned} \underset{\mathbf{E}}{\text{minimize}} \quad & -\mathbb{E}_{(\mathbf{x},y)\sim p_d,\xi}\left[\log \tilde{p}_m(y|x, \xi; \theta, \mathbf{E})\right] \\ & + \lambda \max\left(\log\left(\sum_{l=1}^{L} E^{(l)} n_{mac}^{(l)}\right) - \log\left(E_{\max}\right), 0\right) \end{aligned} \quad (12)$$

This objective can now directly be optimized with respect to the energy per layer via stochastic gradient descent. To train the energy allocations, we can use the Monte Carlo estimator of the objective by sampling data and noise. Because this minimization is performed only over $\mathbf{E}$, not the parameters θ, the optimization problem can typically be solved with a small number of gradient steps on a small subset of the original training dataset. One challenge for the aforementioned method is if the function $\tilde{g}$ includes quantization operations, such as rounding. The gradient of the round function is zero almost everywhere, so the method will not naively be able to learn energy allocations by gradient descent. Following the literature for quantization aware training, we use the Straight Through Estimator (STE) to resolve this issue, effectively computing $\text{grad}_x(\text{round}(x)) = 1$ [54]. Then, dynamic precision can be learned in the presence of deterministic quantization.

While the problem above assumes that energy/MAC is continuous, this method can also be applied when restricted to quantized energy levels, as in the case of redundant coding. This can be done by rounding the energy/MAC to the nearest quantized energy level during training using the STE.

Energy can also be allocated to computations at a finer grained scale than each layer, such as for each channel of a convolutional neural network or each row of a weight matrix. Since each weight channel is convolved over the entire input image, it is reasonable to have dynamic precision by channel, as is done in digital approaches [13]. In this case, we learn $E^{(l,i)}$, or the energy/MAC for the i^{th} channel in layer (l).

VI. Simulations

A. Setup

We evaluate the impacts of dynamic precision on computer vision models and natural language processing models. For computer vision models, we evaluate five image classification models, Resnet50 [55], Mobilenetv2 [56], Inceptionv3 [57], Googlenet [58], and Shufflenetv2 [59], on the ImageNet dataset [60]. For natural language models, we evaluate BERT, a popular transformer architecture [61], fine-tuned for the

TABLE II
MINIMUM ENERGY/MAC WITH <2% ACCURACY DEGRADATION

		Resnet50	Mobilenet	Inceptionv3	Googlenet	Shufflenetv2
	MACs (G)	4.12	0.32	2.85	1.51	0.15
Shot Noise Energy/MAC (aJ)	Uniform	25.4	62.3	39.0	31.5	72.0
	Dynamic Per Layer	4.5	21.2	8.7	10.4	24.0
	Dynamic Per Channel	2.8	15.3	4.8	6.5	18.3
	Improvement (Per Channel)	89.0%	75.4%	87.7%	79.4%	74.6%
Thermal Noise Energy/MAC (relative)	Uniform	36.5	2812	52.3	47.8	1369
	Dynamic Per Layer	13.5	333.5	29.7	27.7	182.2
	Dynamic Per Channel	8.1	122.0	16.1	18.1	110.7
	Improvement (Per Channel)	77.8%	95.7%	69.2%	62.1%	91.9%
Weight Noise Energy/MAC (relative)	Uniform	131.0	1027	316.2	198.1	500.0
	Dynamic Per Layer	48.8	296.2	149.2	115.5	295.1
	Dynamic Per Channel	37.2	263.1	122.5	113.7	241.3
	Improvement (Per Channel)	71.6%	74.4%	61.3%	42.6%	51.7%

GLUE MNLI entailment task [62]. Unless otherwise noted, we assume energy/MAC may be continuously varied.

We evaluate computer vision models subject to three different noise sources: shot noise, thermal noise, and weight noise, using (8)–(10). For shot noise applied to homodyne photoelectric multipliers, we report absolute optical energy consumption in aJ, using a photon energy of 128zJ at $\lambda = 1.55\mu m$ and a photodetector responsivity of $\rho = 1$ [23]. Inputs and weights are continuous-valued, as in neuromorphic computing. For thermal noise and weight noise, which are dependent on architectural implementation and engineering details captured by the parameters σ_t, σ_w, we report the energy/MAC as a relative, unitless quantity. Inputs and weights are digital, using 8-bit uniform quantization. Quantization parameters are calibrated on a small subset of the training data [11]. Because thermal noise variance is dependent on the range of each layer, we clip activations at the 99.99th percentile, following [63], [64]. This improves noise tolerance, similar to adaptive clipping for resistive crossbar arrays [65]. Additional experimental details are in Appendix A, and additional experimental results, such as on the impact of percentile clipping, are in Appendix B. Code is open-sourced at https://github.com/sahajgarg/low_precision_nn.

B. Results

We report the minimum attainable energy/MAC with $<2\%$ accuracy degradation for computer vision models when using uniform precision, dynamic precision per layer, and dynamic precision per channel in Table II. We find that using dynamic precision within a neural network can reduce energy consumption of a single model by 43-96%. The largest improvement is observed for MobilenetV2 subject to thermal noise because Mobilenet is especially sensitive to lowered bit precision [12], and thus requires substantially higher energy/MAC for certain layers. Results are consistent across all three noise sources. We show the tradeoff between optical energy/MAC and accuracy in Fig. 4, including when energy/MAC for each layer is constrained to be a discrete number of photons. The constraint on quantized energy levels does not noticeably affect results.

These results suggest that larger computer vision models may actually be more computationally efficient than smaller computer vision models. Resnet50 obtains the lowest energy/MAC

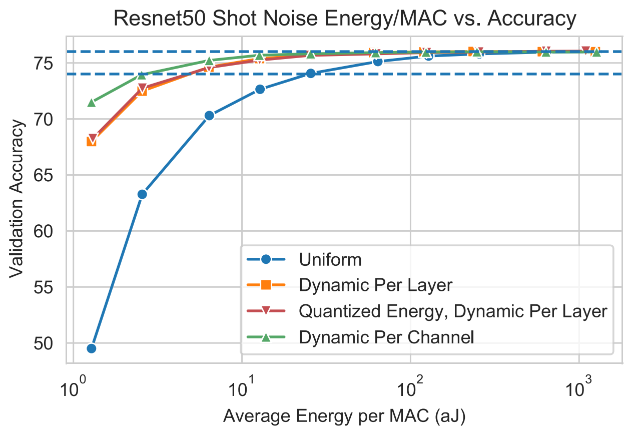


Fig. 4. Accuracy improves with energy/MAC, which reduces the impact of noise. Utilizing dynamic precision improves the allocation of energy to different layers and consequently inference accuracy.

in Table II, likely because it is the largest of the five models in terms of number of MACs, and consequently is the most overparameterized. We evaluate the total energy consumption of Resnet50 and MobilenetV2 when using dynamic precision for Resnet50 to obtain the same accuracy as MobilenetV2. In this case, Resnet50 requires just 0.997 aJ/MAC optical energy consumption, and despite performing 13.6x more MACs than MobilenetV2, consumes 11% less total optical energy. As suggested in [23], this reinforces the importance of designing energy efficient architectures, and not necessarily compressed or small architectures.

We show that the relationship between analog noise and ENOB still holds when using dynamic energy/MAC in Table III. Comparing the rows in Table I and Table III, which correspond to the same average energy/MAC, we observe that the average ENOB for uniform and dynamic precision is similar, but the accuracy of the dynamic precision model is higher because it more effectively allocates bits to precision-sensitive layers. This demonstrates why we measure ENOB per layer. We show the ENOB per layer when using dynamic precision in Fig. 5, and find that the first several and last layer are executed at higher effective bit precision, unlike when using fixed energy/MAC in Fig. 2.

TABLE III
DYNAMIC PRECISION WITH THERMAL NOISE AND ENOB FOR RESNET50

Average Energy/MAC	Noisy Accuracy	Average ENOB	Low-Bit Accuracy
2	51.1	3.2	43.2
5	70.8	3.8	68.3
10	73.7	4.3	73.0
20	74.7	4.8	74.3
29	74.9	5.1	74.7
39	75.0	5.3	75.0
50	75.2	5.5	75.1
99	75.3	6.0	75.2
196	75.4	6.4	75.3
488	75.4	7.1	75.4

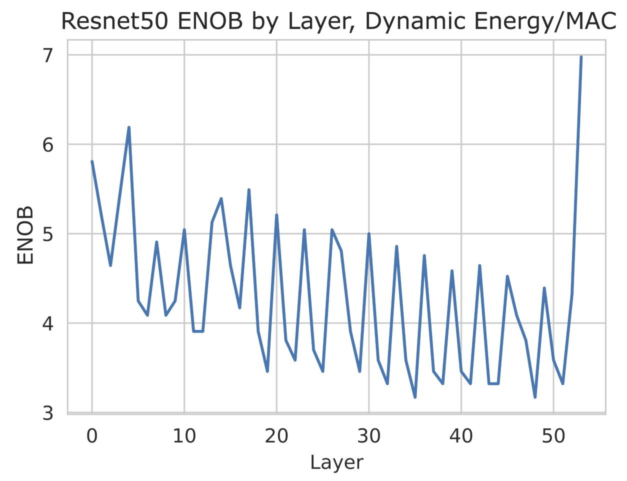


Fig. 5. Effective number of bits of precision when using dynamic energy/MAC for different layers of Resnet50, average ENOB 4.8.

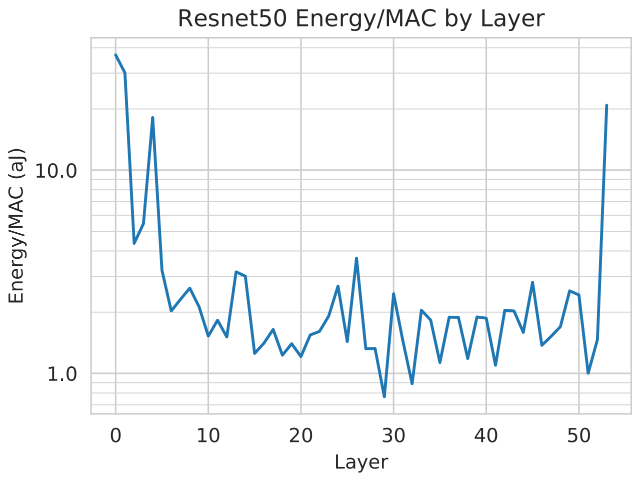


Fig. 6. Allocations of energy to each layer are complex. The first several and last layer are allocated higher energy/MAC, and the allocations follow sawtooth patterns due to the 3-layer building blocks of the network architecture.

We examine the dynamic energy allocations per layer of Resnet50 in Fig. 6 to better understand why dynamic precision improves performance. The energy/MAC varies substantially by layer: the first few and last layers require >10x the energy/MAC of other layers. We infer that if these layers are not run at high precision, the neural network will not produce accurate results. Hence, using uniform precision per layer will require using the energy/MAC needed for the most sensitive layer. We further observe that the final energy allocations are complex. This is similar to the observations on mixed precision inference for digital neural networks [15], [16], [17], [21]. These observations emphasize the importance of using an empirical, automatic method instead of manually or analytically determining the required precision.

TABLE IV
BERT SHOT NOISE CONSTRAINED ENERGY/MAC (aJ)

	MNLI Dataset
Uniform	2.1
Dynamic Per Layer	1.6
Improvement	24%

Finally, we evaluate the minimum energy/MAC required for BERT inference on the MNLI Dataset in Table IV. Using dynamic precision per layer improves BERT energy consumption by 24% to just 1.6 aJ/MAC, lower than any of the computer vision models. The energy improvement for using dynamic precision in BERT is smaller than for computer vision models. This is likely because no large layers in BERT require high precision, unlike the first and last layer of computer vision models, which are precision sensitive and perform around 20% of the total MACs in the network. The energy/MAC for different matrix multiplications in BERT is reported in Appendix B. Regardless, dynamic precision in analog hardware is necessary to enable BERT inference at low energy/MAC while also allowing for the higher energy settings required by computer vision models.

VII. DISCUSSION

These results emphasize the importance of designing analog computing architectures that can support programmable and dynamic precision. The required energy/MAC when using uniform precision for different models can range from 2.1 aJ/MAC for BERT to 72 aJ/MAC for ShufflenetV2. If analog architectures do not support dynamic precision, then new neural networks may require higher precision than is supported, and render existing hardware nonfunctional. Moreover, enabling dynamic precision allows programmers to develop novel techniques for lowering the energy requirements of analog neural networks. This work demonstrates one such technique for utilizing dynamic precision for different layers or channels of neural networks to reduce energy consumption by 43–96%.

By enabling dynamic precision, this work demonstrates that the bound on optical energy consumption of homodyne photoelectric multipliers, set by shot noise in photodetectors, can be as low as 1.6 aJ/MAC. This extends the result in [23] for MLPs and shallower convolutional networks like AlexNet to deeper models such as Resnet50, for which the bound on energy/MAC is as low as 2.7 aJ/MAC. We note that this energy consumption is for an idealized system with no optical loss in the shot noise limited regime, and does not account for energy consumption of data movement, analog to digital conversion, or memory traffic. These figures are primarily meant as a demonstration that dynamic precision can be used to prevent optical matrix multiplication energy expenditure from being the system bottleneck. To more accurately capture the total energy consumption, the framework can be extended to incorporate ADC energy as a cost that scales linearly with redundant coding, but that does not affect noise for the layer, allowing more appropriate tradeoffs to be made. Other costs such as memory access and data movement cost can be similarly modeled.

This work enables dynamic precision through a relatively straightforward technique, redundant coding, which has substantial tradeoffs throughput or compute density. In practice, this technique should not be used at it's most extreme, but rather to increase the precision of a handful of layers that require higher precision. Other architecture-specific methods for enabling dynamic precision may incur less steep penalties in other important metrics.

In addition to varying the precision in analog computing resulting from noise, it is possible to vary the bit precision of analog architectures that use digital inputs, weights, and/or outputs. When using dynamic precision, many inputs and weights may be subject to noise of sufficiently large magnitude that several of the least significant bits are discarded. We see this in Fig. 5, where some layers of Resnet50 use fewer than 4 effective bits but are quantized to 8 bit integers. It may be possible to dynamically set analog-to-digital converter precision based on the number of bits of noise precision in different computations. Moreover, the optimization problem in (12) can be extended to jointly learn the optimal number of bits to allocate per layer, as done by [21] for digital architectures.

Depending on the hardware architecture, the total energy penalty may need to be modified. In this work, we only model the energy consumed by the matrix multiplier; however, substantial energy is consumed by data movement, portions of which may be done digitally, or by other operations such as partial sum accumulation or nonlinearities [1]. Approaches such as time averaging only require moving more data if the bit precision of inputs is increased, so the energy/MAC may not scale linearly with the amount of red‘undancy if data movement is modeled. If the bit precision of different layers is also learned, the energy penalty may also include memory pressure based on the bitwidth of activations and weights, analog-to-digital converter energy consumption, and other factors.

Finally, even more energy efficient models may be enabled by training neural networks to be more noise tolerant. Many approaches to obtaining low bit precision require retraining the network parameters while simulating the effect of quantization to obtain high accuracy [12], [15], [21]. A similar noise-aware training process [66] can be applied to jointly learning network parameters and dynamic precision allocations by modifying the optimization problem in (12) to also optimize over the parameters θ of the neural network and other quantizer parameters such as the range of different layers (as in [67]), which affect noise magnitude. In this regime, optical computers will not suffer energy limitations due to optical energy consumption, but rather due to analog to digital conversions, memory accesses, and other bottlenecks.

VIII. Conclusion

In this work, we demonstrate the utility of extending analog computing architectures to support dynamic precision with redundant coding. By repeating operations and averaging the result, redundant coding enables programmable tradeoffs between precision and other desirable performance metrics, such as energy efficiency or throughput. Enabling dynamic precision is critical for supporting different models that require different precision: for example, Shufflenetv2 requires 3x the energy/MAC of Resnet50, and 34x the energy/MAC of BERT. Moreover, we show that it is possible to leverage dynamic precision within a single model by solving an optimization problem to maximize log likelihood while adding a penalty for energy consumption, and that using dynamic precision within a model improves energy consumption by 43-96%. In one example of optical neural networks limited by shot noise, dynamic precision enables Resnet50 inference at an optical energy consumption of just 2.7 aJ/MAC and BERT at 1.6 aJ/MAC with $<2\%$ accuracy degradation. These results emphasize the importance of designing analog architectures to support dynamic precision.

Appendix A
Simulation Details

We provide additional details for the experimental setup. Code and scripts for generating results are open-sourced at https://github.com/sahajgarg/low_precision_nn.

Noise: For thermal noise, we set $\sigma_t = 0.01$ and for weight noise, we set $\sigma_w = 0.1$. These choices were arbitrary, so energy/MAC is referred to as a relative quantity in the main text. For specific architectures, they should be measured. We assume that residual connections, concatenation operations, max pooling, and average pooling occur without additional noise. We restrict our evaluation for BERT to shot noise because self-attention layers of BERT, which multiply two activation matrices, may be challenging to compute in-memory.

Quantization: We use per-channel quantization of weights and per-tensor quantization of activations [12]. For linear layers, we perform quantization per row of the weight matrix, analogous to per-channel quantization of convolutional layers. Weight quantization parameters are determined by calibration on the min/max values of the weights in each channel. When evaluating subject to thermal noise, the minimum and maximum values for activations are set based on the 99.99th percentile of the data, evaluated over 120 training examples [63], [64]. Because percentile clipping degrades accuracy by 0.3% when activations are at high precision, but improves accuracy at lower activation precision (shown in Appendix B), it is used only for thermal noise. A better strategy for clipping parameters may be to learn the quantizer minimum and maximum, as in [67], jointly with energy allocations, which we leave to future work. For weight noise, activations are calibrated based on a moving average of the min/max values of the data over 100 batches with a batch size of 32. The skip and residual connections are quantized, and the outputs are requantized to 8 bits.

Training: We train optimal energy allocations $\mathbf{E}$ using 4% of the training dataset for one epoch, which takes <10 minutes

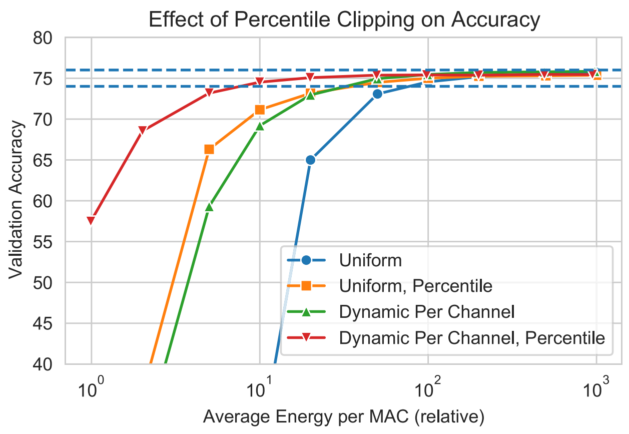


Fig. 7. Percentile clipping of activations improves accuracy subject to thermal noise.

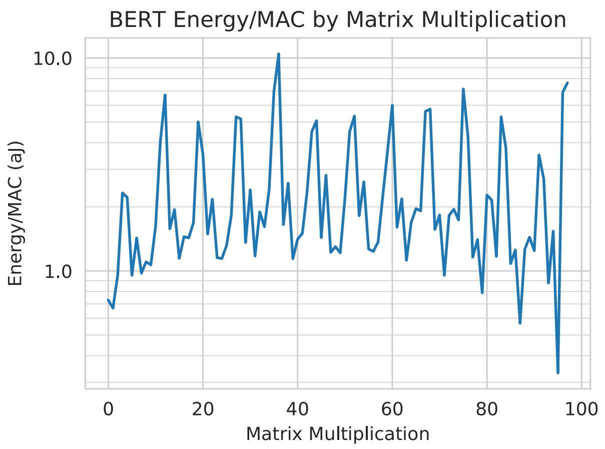


Fig. 8. Bert Energy/MAC for different matrix multiplications in BERT. Note that each layer performs multiple matrix multiplications. While some layers utilize more energy/MAC, such as the last layer, these layers perform an extremely small fraction of the total MACs in the network and consequently do not contribute substantially to the total energy consumption.

on a NVIDIA V100 GPU, a small fraction of the time to train the entire model. Energy allocations are trained using the Adam optimizer with a learning rate of 0.01 [68]. The penalty hyperparameter was set to $\lambda = 2$ for shot noise, and $\lambda = 8$ for thermal and weight noise; we found that the method was relatively insensitive to the choice of λ. We did not do extensive experimentation with respect to the required dataset size, but found that the method was relatively insensitive to the use of less calibration data than presented in this work, assuming the energy allocations are trained until convergence.

Evaluation: All results are reported on the corresponding validation datasets. For each task, we determine the minimum average energy/MAC for which the accuracy does not degrade below floating point accuracy by 2% (within 0.1%) by performing a binary search on the target energy/MAC. Accuracy degradation of all models except Mobilenet is measured with respect to the respective floating point baseline. For Mobilenet, 8 bit quantization degrades accuracy by $>1\%$, so we evaluate 2% accuracy degradation relative to the 8 bit baseline. For BERT evaluation, the MNLI entailment task reports both matched and mismatched accuracy on entailment. We measure the average accuracy degradation of the two metrics.

Appendix B
Additional Results

In Fig. 7, we evaluate the impact of percentile clipping on accuracy subject to thermal noise. We find that with high amounts of noise, percentile clipping of activations improves accuracy of both uniform precision and dynamic precision models, and uniform precision with percentile clipping outperforms dynamic precision without clipping. This is likely because the magnitude of thermal noise is proportional to the range of inputs, and clipping at the 99.99th percentile reduces the range by approximately half. However, at high precision, it degrades accuracy by 0.3%.

In Fig. 8, we show the energy/MAC for different matrix multiplications in BERT, and in Fig. 9. we show the energy/MAC for different layers of MobilenetV2.

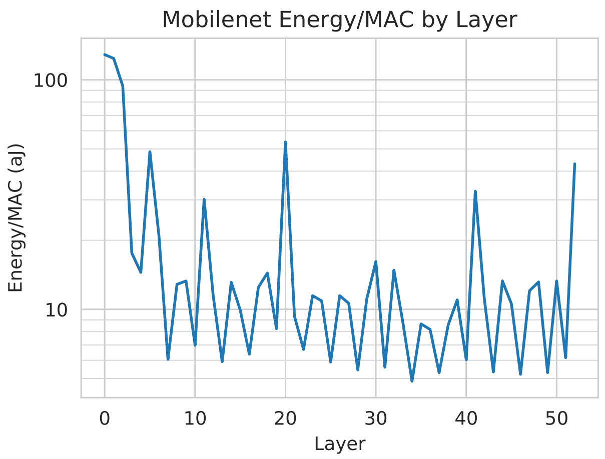


Fig. 9. Allocations of energy to each layer for MobilenetV2 demonstrate similar results to Resnet50.

Acknowledgment

We would like to thank many members of the Luminous Computing team, including Michael Gao, Matthew Chang, Rodolfo Camacho-Aguilera, Rohun Saxena, Katherine Roelofs, and Patrick Gallagher, as well as Michael Hochberg, for their helpful discussions and suggestions.

References

[1] M. A. Nahmias et al., "Photonic multiply-accumulate operations for neural networks," *IEEE J. Sel. Topics Quantum Electron.*, vol. 26, no. 1, pp. 1–18, Jan./Feb. 2020.
[2] L. Fick et al., "Analog in-memory subthreshold deep neural network accelerator," in *Proc. IEEE Custom Integr. Circuits Conf.*, 2017, pp. 1–4.
[3] V. Joshi et al., "Accurate deep neural network inference using computational phase-change memory," *Nature Commun.*, vol. 11, no. 1, pp. 1–13, 2020.
[4] A. Shafiee et al., "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proc. ACM/IEEE 43rd Annu. Int. Symp. Comput. Architecture*, 2016, pp. 14–26.

[5] L. D. Marinis, M. Cococcioni, P. Castoldi, and N. Andriolli, "Photonic neural networks: A survey," *IEEE Access*, vol. 7, pp. 175827–175841, 2019.

[6] B. J. Shastri et al., "Photonics for artificial intelligence and neuromorphic computing," *Nature Photon.*, vol. 15, no. 2, pp. 102–114, Feb. 2021, doi: 10.1038/s41566-020-00754-y.

[7] Y. Shen et al., "Deep learning with coherent nanophotonic circuits," *Nature Photon.*, vol. 11, no. 7, pp. 441–446, 2017.

[8] D. A. B. Miller, "Perfect optics with imperfect components," *Optica*, vol. 2, no. 8, pp. 747–750, Aug. 2015. [Online]. Available: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-2-8-747

[9] M. Miscuglio and V. Sorger, "Photonic tensor cores for machine learning," *Appl. Phys. Rev.*, vol. 7, 2020, Art. no. 031404.

[10] D. Amodei and D. Hernandez, "Ai and compute," 2020. [Online]. Available: https://openai.com/blog/ai-and-compute/

[11] B. Jacob et al., "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 2704–2713.

[12] R. Krishnamoorthi, "Quantizing deep convolutional networks for efficient inference: A whitepaper," 2018, *arXiv:1806.08342*.

[13] R. Banner, Y. Nahshan, and D. Soudry, "Post training 4-bit quantization of convolutional networks for rapid-deployment," in *Proc. Adv. Neural Inf. Process. Syst.*, H. Wallach et al., Eds., vol. 32, 2019, pp. 7950–7958. [Online]. Available: https://arxiv.org/abs/1810.05723

[14] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer, "HAWQ: Hessian aware quantization of neural networks with mixed-precision," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 293–302.

[15] Z. Dong et al., "HAWQ-V2: Hessian aware trace-weighted quantization of neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 18518–18529, 2020.

[16] K. Wang, Z. Liu, Y. Lin, J. Lin, and S. Han, "HAQ: Hardware-aware automated quantization with mixed precision," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 8604–8612.

[17] B. Wu et al., "Mixed precision quantization of convnets via differentiable neural architecture search," 2018, *arXiv:1812.00090*.

[18] Y. Cai et al., "Zeroq: A novel zero shot quantization framework," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 13166–13175.

[19] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," vol. abs/2006.10518, 2020.

[20] D. Lin, S. Talathi, and S. Annapureddy, "Fixed point quantization of deep convolutional networks," in *Proc. 33rd Int. Conf. Mach. Learn.*, M. F. Balcan and K. Q. Weinberger, Eds., vol. 48. New York, New York, USA, 2016, pp. 2849–2858. [Online]. Available: http://proceedings.mlr.press/v48/linb16.html

[21] S. Uhlich et al., "Mixed precision DNNs: All you need is a good parametrization," in *Proc. Int. Conf. Learn. Representations*, 2020, pp. 67–68.

[22] NVIDIA, "NVIDIA A100 tensor core GPU architecture," Tech. Rep., 2020. [Online]. Available: https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/nvidia-ampere-architecture-whitepaper.pdf

[23] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, and D. Englund, "Large-scale optical neural networks based on photoelectric multiplication," *Phys. Rev. X*, vol. 9, May 2019, Art. no. 021032.

[24] I. Boybat et al., "Neuromorphic computing with multi-memristive synapses," *Nature Commun.*, vol. 9, no. 1, pp. 1–12, 2018.

[25] M. Gallo et al., "Mixed-precision in-memory computing," *Nature Electron.*, vol. 1, pp. 246–253, 2018.

[26] S. R. Nandakumar et al., "Mixed-precision architecture based on computational memory for training deep neural networks," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2018, pp. 1–5.

[27] E. Eleftheriou et al., "Deep learning acceleration based on in-memory computing," *IBM J. Res. Dev.*, vol. 63, no. 6, pp. 7:1–7:16, 2019.

[28] S. R. Nandakumar et al., "Mixed-precision deep learning based on computational memory," *Front. Neurosci.*, vol. 14, 2020, Art. no. 406. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2020.00406

[29] G. W. Burr et al., "Neuromorphic computing using non-volatile memory," *Adv. Phys.: X*, vol. 2, no. 1, pp. 89–124, 2017, doi: 10.1080/23746149.2016.1259585.

[30] J. J. Yang, D. Strukov, and D. Stewart, "Memristive devices for computing," *Nature Nanotechnol.*, vol. 8, pp. 13–24, 2013.

[31] H.-Y. Tsai, S. Ambrogio, P. Narayanan, R. Shelby, and G. Burr, "Recent progress in analog memory-based accelerators for deep learning," *J. Phys. D: Appl. Phys.*, vol. 51, no. 28, 2018, Art. no. 283001.

[32] C. Li et al., "Long short-term memory networks in memristor crossbar arrays," *Nature Mach. Intell.*, vol. 1, no. 1, pp. 49–57, 2019.

[33] Z. Wang et al., "Reinforcement learning with analogue memristor arrays," *Nature Electron.*, vol. 2, no. 3, pp. 115–124, 2019.

[34] S. Agarwal et al., "Energy scaling advantages of resistive memory crossbar based computation and its application to sparse coding," *Front. Neurosci.*, vol. 9, 2016, Art. no. 484. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2015.00484

[35] M. R. Mahmoodi, M. Prezioso, and D. B. Strukov, "Versatile stochastic dot product circuits based on nonvolatile memories for high performance neurocomputing and neurooptimization," *Nature Commun.*, vol. 10, no. 1, Nov. 2019, Art. no. 5113, doi: 10.1038/s41467-019-13103-7.

[36] A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Broadcast and weight: An integrated network for scalable photonic spike processing," *J. Lightw. Technol.*, vol. 32, no. 21, pp. 4029–4041, Nov. 2014.

[37] T. F. de Lima et al., "Machine learning with neuromorphic photonics," *J. Lightw. Technol.*, vol. 37, no. 5, pp. 1515–1534, Mar. 2019. [Online]. Available: http://jlt.osa.org/abstract.cfm?URI=jlt-37-5-1515

[38] H. Peng, M. A. Nahmias, T. F. de Lima, A. N. Tait, and B. J. Shastri, "Neuromorphic photonic integrated circuits," *IEEE J. Sel. Topics Quantum Electron.*, vol. 24, no. 6, pp. 1–15, Nov./Dec. 2018.

[39] L. Yang, R. Ji, L. Zhang, J. Ding, and Q. Xu, "On-chip cmos-compatible optical signal processor," *Opt. Exp.*, vol. 20, no. 12, pp. 13560–13565, Jun. 2012. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-20-12-13560

[40] V. Bangari et al., "Digital electronics and analog photonics for convolutional neural networks (DEAP-CNNs)," *IEEE J. Sel. Topics Quantum Electron.*, vol. 26, no. 1, pp. 1–13, Jan./Feb. 2020.

[41] A. N. Tait et al., "Silicon photonic modulator neuron," *Phys. Rev. Appl.*, vol. 11, Jun. 2019, Art. no. 064043, doi: 10.1103/PhysRevApplied.11.064043.

[42] A. N. Tait et al., "Microring weight banks," *IEEE J. Sel. Topics Quantum Electron.*, vol. 22, no. 6, pp. 312–325, Nov./Dec. 2016.

[43] A. Tait et al., "Neuromorphic photonic networks using silicon photonic weight banks," *Sci. Rep.*, vol. 7, no. 1, pp. 1–10, 2017.

[44] M. A. Nahmias et al., "A laser spiking neuron in a photonic integrated circuit," 2020.

[45] M. Hayat, B. Saleh, and J. Gubner, "Shot-noise-limited performance of optical neural networks," *IEEE Trans. Neural Netw.*, vol. 7, no. 3, pp. 700–708, May 1996.

[46] S. Agarwal et al., "Resistive memory device requirements for a neural algorithm accelerator," in *Proc. Int. Joint Conf. Neural Netw.*, 2016, pp. 929–938.

[47] M. Hu et al., "Dot-product engine for neuromorphic computing: Programming 1T1M crossbar to accelerate matrix-vector multiplication," in *Proc. 53nd ACM/EDAC/IEEE Des. Automat. Conf.*, 2016, pp. 1–6.

[48] M. A. Nahmias, "System for photonic computing," U. S. Patent 10,656,336, Apr. 2020.

[49] B. Feinberg, S. Wang, and E. Ipek, "Making memristive neural network accelerators reliable," in *Proc. IEEE Int. Symp. High Perform. Comput. Architecture*, 2018, pp. 52–65.

[50] H. Sim, S. Kenzhegulov, and J. Lee, "DPS: Dynamic precision scaling for stochastic computing-based deep neural networks*," in *Proc. 55th ACM/ESDA/IEEE Des. Automat. Conf.*, 2018, pp. 1–6.

[51] *IEEE Standard for Terminology and Test Methods for Analog-to-Digital Converters*, IEEE Std 1241-2010 (Revision of IEEE Std 1241-2000), 2011.

[52] A. Sripad and D. Snyder, "A necessary and sufficient condition for quantization errors to be uniform and white," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 25, no. 5, pp. 442–448, 1977.

[53] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods* (Optimization and Neural Computation Series), 1st ed. Nashua, NH, USA: Athena Scientific, 1996.

[54] Y. Bengio, N. Léonard, and A. C. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," 2013, *arXiv:1308.3432*.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 770–778.

[56] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 4510–4520.

[57] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 2818–2826.

[58] C. Szegedy et al., "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1–9.

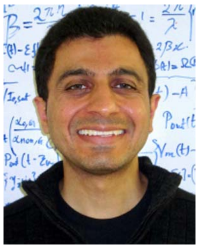
[59] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet V2: Practical guidelines for efficient CNN architecture design,” in *Proc. Eur. Conf. Comput. Vis.*, 2018, pp. 116–131.
[60] J. Deng et al., “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[61] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol., Volume 1 (Long Short Papers)*, Minneapolis, Minnesota, 2019, pp. 4171–4186. [Online]. Available: https://www.aclweb.org/anthology/N19-1423
[62] A. Wang et al., “GLUE: A multi-task benchmark and analysis platform for natural language understanding,” in *Proc. EMNLP Workshop BlackboxNLP: Analyzing Interpreting Neural Netw. NLP*, Brussels, Belgium, 2018, pp. 353–355. [Online]. Available: https://www.aclweb.org/anthology/W18-5446
[63] R. Li et al., “Fully quantized network for object detection,” in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 2805–2814.
[64] N. Ma, X. Zhang, H.-T. Zheng, and J. Sun, “Shufflenet v2: Practical guidelines for efficient cnn architecture design,” in *Proc. European Conf. Comput. Vis.*, Sep. 2018.
[65] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.
[66] A. S. Rekhi et al., “Analog/mixed-signal hardware error modeling for deep learning inference,” in *Proc. 56th Annu. Des. Autom. Conf.*, New York, NY, USA, 2019, pp. 1–6. [Online]. Available: https://doi.org/10.1145/3316781.3317770
[67] S. K. Esser, J. L. McKinstry, D. Bablani, R. Appuswamy, and D. S. Modha, “Learned step size quantization,” in *Proc. Int. Conf. Learn. Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rkgO66VKDS
[68] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *Proc. Int. Conf. Learn. Representations*, 2014.

Sahaj Garg received the B.S. degree in computer science, with a specialization in machine learning, from Stanford University, Stanford, CA, USA, in 2020. He is currently the Co-founder and Chief Technology Officer with Wispr AI, a company building neural interfaces for the next generation of human-computer interaction. From 2018 to 2020, he was a Research Assistant with the Stanford AI Laboratory. From 2020 to 2021, he was the AI Lead with Luminous Computing, where he focused on the interactions between analog computing and machine learning.

Joe Lou received the B.S. degree in electrical engineering and the M.S. degree in computer science from Stanford University, Stanford, CA, USA, in 2020. While at Stanford, he was a Research Assistant with the Ng and Leskovec labs. From 2020 to 2021, he was an ML Engineer with Luminous Computing, where this work was completed. From 2021 to 2022, he was part of the initial team developing algorithmic fairness solutions for recommender models at Facebook, and since 2022 he has been with Wispr AI, a company in South San Francisco creating the software and ML stack for next generation neural interfaces.

Anirudh Jain received the B.S. and M.S. degrees in computer science, with specializations in computer systems and machine learning from Stanford University, Stanford, CA, USA in 2020 and 2021 respectively. From 2019 to 2020, he was a Research Assistant with the Stanford AI Laboratory researching generative modeling. From 2020 to 2021, he was a Senior Machine Learning Engineer with Luminous Computing, where he focused on optimizing machine learning models for photonic-based analog computing architectures. Since 2021, he has been a Senior Machine Learning Engineer with Scale AI, focused on developing the next suite of machine learning infrastructure and operations products.

Zhimu Guo received the B.A.Sc. and M.A.Sc degrees in engineering physics, computing option from Queen’s University, Kingston, ON, Canada, where he is currently working toward the Ph.D. degree. He has a strong interest in working with the junction of the hardware and software for computer systems. He is also looking forward to exploring new technologies in the quantum computing realm, including integrated neuromorphic photonic processors for deep learning.



Bhavin J. Shastri (Senior Member, IEEE) received the Ph.D. degree in electrical engineering (photonics) from McGill University, Montreal, QC, Canada, in 2012. He is currently an Assistant Professor of engineering physics with Queen’s University, Kingston, ON, Canada, and a Faculty Affiliate with the Vector Institute for Artificial Intelligence, Canada. He was an Associate Research Scholar (2016–2018) and Banting/NSERC Postdoctoral Fellow (2012–2016) with Princeton University, Princeton, NJ, USA. He has authored more than 70 journal articles and 100 conference proceeding, seven book chapters, and given more than 65 invited talks and lectures including five keynotes and three tutorials. He is a co-author of the book (CRC Press, 2017) *Neuromorphic Photonics*, a term he helped coin. His research interests include silicon photonics, photonic integrated circuits, neuromorphic computing, and machine learning.Dr. Shastri was the recipient of the 2022 SPIE Early Career Achievement Award and the 2020 IUPAP Young Scientist Prize in Optics for his pioneering contributions to neuromorphic photonics from ICO, 2014 Banting Postdoctoral Fellowship from the Government of Canada, the 2012 D. W. Ambridge Prize for the top graduating Ph.D. student at McGill, an IEEE Photonics Society 2011 Graduate Student Fellowship amongst others awards. He is a Senior Member of Optica (formerly OSA).

Mitchell Nahmias received the B.S., M.A., and Ph. D. degrees from Princeton University, Princeton, NJ, USA. He is a Co-founder and Chief Technology Officer of Luminous Computing. While at Princeton, he was a contributing author to the textbook on *Neuromorphic Photonics*, and is an author on more than 70 papers and patents. He was the recipient of the NSF Graduate Research Fellowship by Princeton University.