IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 22 June 2022, accepted 14 July 2022, date of publication 20 July 2022, date of current version 8 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3192515

RESEARCH ARTICLE

ENOS: Energy-Aware Network Operator Search in Deep Neural Networks

SHAMMA NASRIN[1], (Graduate Student Member, IEEE), AHISH SHYLENDRA[1], (Member, IEEE), NASTARAN DARABI[1], (Graduate Student Member, IEEE), THEJA TULABANDHULA[2], WILFRED GOMES[3], ANKUSH CHAKRABARTY[4] (Senior Member, IEEE), AND AMIT RANJAN TRIVEDI[1], (Senior Member, IEEE)

[1]Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL 60607, USA
[2]Department of Information and Decision Sciences, University of Illinois at Chicago, Chicago, IL 60607, USA
[3]Intel Corporation, Hillsboro, OR 97124, USA
[4]Mitsubishi Electric Research Laboratories, Cambridge, MA 02139, USA

Corresponding author: Shamma Nasrin (snasri2@uic.edu)

ABSTRACT This work proposes a novel *Energy-aware Network Operator Search (ENOS)* approach to address the energy-accuracy trade-offs of a deep neural network (DNN) accelerator. In recent years, novel hardware-friendly inference operators such as binary-weight, multiplication-free, and deep-shift have been proposed to improve the computational efficiency of a DNN accelerator. However, simplifying DNN operators invariably comes at lower accuracy, especially on complex processing tasks. While prior works generally implement the same inference operator throughout the neural architecture, the proposed ENOS framework allows an *optimal layer-wise integration of inference operators with optimal precision* to maintain high prediction accuracy and high energy efficiency. The search in ENOS is formulated as a continuous optimization problem, solvable using gradient descent methods, thereby minimally increasing the training cost when learning *both* layer-wise inference operators and weights. Utilizing ENOS, we discuss multiply-accumulate (MAC) cores for digital spatial architectures that can be reconfigured to different operators and varying computing precision. ENOS training methods with single and bi-level optimization objectives are discussed and compared. We also discuss a sequential operator assignment strategy in ENOS that only learns the assignment for one layer in one training step. Furthermore, a stochastic mode of ENOS is also presented. ENOS is characterized on ShuffleNet and SqueezeNet using CIFAR10 and CIFAR100. Compared to the conventional uni-operator approaches, under the same energy budget, ENOS improves accuracy by 10–20%. ENOS also outperforms the accuracy of comparable mixed-precision uni-operator implementations by 3-5% for the same energy budget.

INDEX TERMS Low power, deep neural network, mixed-precision learning.

I. INTRODUCTION

The proliferation of deep learning in embedded computing spaces such as computer vision [1], cyber security [2], predictive maintenance [3], autonomous drones [4], *etc.*, with limited processing and storage resources has created an imminent need for computationally efficient deep neural networks (DNNs). DNN architectures are going through a dramatic evolution to improve their prediction capacity while operating under stringent memory and processing constraints. Most DNN architectures are designed by domain experts who approach each architecture on a case-by-case basis, depending on the application requirements and computing substrate; however, the resultant design approaches

The associate editor coordinating the review of this manuscript and approving it for publication was Oussama Habachi.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

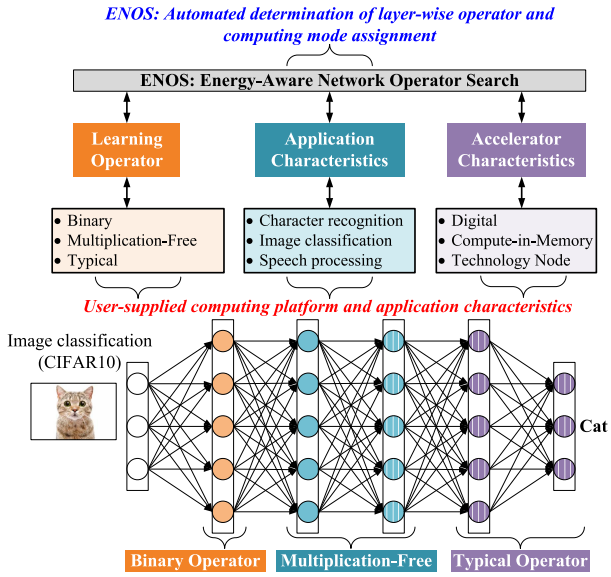

FIGURE 1. ENOS searches for optimal layer-wise operators to maximize energy efficiency of inference in a DNN while adhering to application and computing platform characteristics such as technology node.

are typically not scalable, and their efficacy is highly subjective. Conversely, neural architecture search (NAS) algorithms can automate the design procedure, providing a scalable, optimization-based design framework that does not require the significant involvement of human experts. Among recent NAS approaches, NASnet [5] learns an optimal convolutional neural architecture on a smaller dataset using reinforcement learning and then transfers the learned architecture to a larger dataset. In Efficient Neural Architecture Search (ENAS) [6], a controller is trained using policy gradient to find an optimal subgraph within a large computational graph. Unlike prior reinforcement learning-based approaches, which learn over a discrete search space and attempt to find a globally optimal network configuration, but require thousands of GPU hours to compute, in [7], architecture search was solved using gradient descent by formulating the search space to be continuous. Using [7], a locally optimal architecture search could be performed within a few GPU hours on modern DNNs.

In parallel, considerable efforts have also optimized DNN's inference operators. Scalar multiplication of input and weight vectors is the most prevalent inference operator in DNNs. Since DNN's processing involves evaluating thousands to millions of scalar multiplications among high-dimensional input/weight vectors, the operator dictates the overall computational efficiency of a DNN. Therefore, prior works have explored alternate, more computationally efficient inference operators. [8], [9] introduced multiplication-free operation on compute-in-memory hardware where high-precision multiplication between input and weight vectors is replaced by 1-bit multiplication and multi-bit addition. [10] introduced bit-wise shift operators where weights are approximated by a power of two, i.e., 2^n ($n \in \mathbb{Z}^+$), replacing multiplications with shift operations. [11], [12] quantize inputs and weights to one-bit and uses XNOR instead of multiplications. The new inference operators can further benefit from custom-designed computing modes. For example, multiplication-free and binary weight operators are quite suited for compute-in-memory [8], [13], [14]. A deep-shift operator can be more efficiently implemented using digital shifters [10]. Operators in [11], [12] can be efficiently implemented using XOR logic gates. Optimally combining inference operators with an optimal computing mode can dramatically improve the energy efficiency of DNNs.

Our novel framework on *energy-aware network operator search* (ENOS) synergistically integrates the above two efforts, namely, neural network design automation and exploration of more computationally efficient learning operators. ENOS automatically combines various correlation operators layer-wise so that energy-accuracy trade-offs in a DNN can be optimally balanced. For example, unlike the current approaches, which consider end-to-end DNN processing using a single correlation operator, in Figure 1, ENOS can optimally integrate typical, binary weight, and multiplication-free operators among DNN layers. Thereby, DNN layers with less impact on network accuracy can be opportunistically operated with simplified operators while layers of higher impact to accuracy can be kept to computationally complex traditional operators. The presented framework of ENOS is also able to recognize the critical hardware characteristics such as computing energy on a unit weight-input correlation and the energy to retrieve weights/inputs under the accelerator's memory organization and can account for them when assigning optimal layer-wise deep learning operators. Therefore, the automated synthesis in ENOS can inherently depend on application and hardware characteristics. Furthermore, inspired by [7], ENOS learns in a continuous search space using standard gradient descent-based routines (such as ADAM [15]) for scalable training on complex DNNs.

In Sec. II, we discuss the overview of ENOS. In Sec. III, we characterize ENOS on digital platforms while showing results using single-level, bi-level, and sequential optimization strategies and a stochastic search and design perspective. Sec. IV provides concluding remarks.

II. ENERGY-AWARE NETWORK OPERATOR SEARCH (ENOS) FRAMEWORK

Prior works have explored novel deep learning operators such as binarized weights and inputs [11], binary-connect [12], multiplication-free [8], and deep-shift [10] to improve the computational efficiency of deep learning models within limited computing and storage resources. However, the computational advantages of simplified operators come at the cost of accuracy degradation. Moreover, the efficiency of novel operators also strongly depends on hardware and task specifications. For example, deep networks based on binarized weights and inputs can be very efficiently implemented using a network of XOR logic gates [16], [17]. Despite showing a competitive accuracy on datasets such as MNIST [18], the

operator has limited accuracy on more complex datasets such as ImageNet [19] compared to the typical case, i.e., full precision scalar product operator. Therefore, a unique opportunity exists where these operators can be optimally combined to harness their computational efficiency while maintaining high prediction accuracy. Moreover, such optimal layer-wise fusion of diverse network operators should also consider the underlying hardware characteristics, energy constraints, and task specifications. We present a systematic framework for such automated operator allocation to address this challenge.

A. SINGLE-LEVEL AND BI-LEVEL ENOS

Figure 2 shows the framework of ENOS, which is inspired from [7]. We consider an optimal operator choice among typical, binary-connect, and multiplication-free operators at each layer for the ensuing discussion. However, our approach is generalizable to any number of different operator choices. The input feature map is processed against all three operators across three parallel paths at each layer. For layer input, x_i at layer i, the net output $\overline{y_i}$ is computed by combining output from all three operators as

$$\overline{y_i} = \sum_{j=1}^{N} \text{softmax}(\alpha_{ij}) y_{ij}(x_i). \tag{1}$$

Here, $y_{ij}(x_i)$ is the output computed by the operator j and α_{ij} is the corresponding continuously varying weight factor which is a learning parameter itself. The overall cost function of DNN is, thus, defined as

$$\mathcal{L}_{\text{net}} = \underbrace{\mathcal{L}_{\text{acc}}(\theta, \alpha)}_{\text{Accuracy}} + \lambda \underbrace{\sum_{i=1}^{N} N_{OP,i} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{OP,j}}_{\text{Regularizer for computing energy}}$$
$$+ \lambda \underbrace{\sum_{i=1}^{N} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{TX,ij}}_{\text{Regularizer for transmission energy}}, \tag{2a}$$

$$\theta^*, \alpha^* = \text{argmin}(\mathcal{L}_{\text{net}}). \tag{2b}$$

Here, the first term of $\mathcal{L}_{\text{net}}$, i.e., $\mathcal{L}_{\text{acc}}(\theta, \alpha)$ minimizes with higher prediction accuracy while the second term minimizes with lower energy of computations. i indexes over network layers, while j indexes over operator choices for each layer. The third term minimizes the lower transmission overhead and is a function of both the layer configuration and layer operator. θ are the network weight parameters. α are the layer-wise operator weight parameters. λ is a user-defined hyper-parameter that considers the energy-accuracy trade-off. $N_{OP,i}$ is the number of multiply-accumulate (MAC) operations at layer i. $E_{OP,j}$ is average energy of an operator j for a unit operation. $E_{TX,ij}$ is the average data transmission energy of an operator j among computing module and storage units on layer i, dependent on accelerator characteristics such as its memory organization.

Without the following non-linear activation (such as ReLU), the operation of input ($\mathbf{x}$) and weight ($\mathbf{w}$) vectors

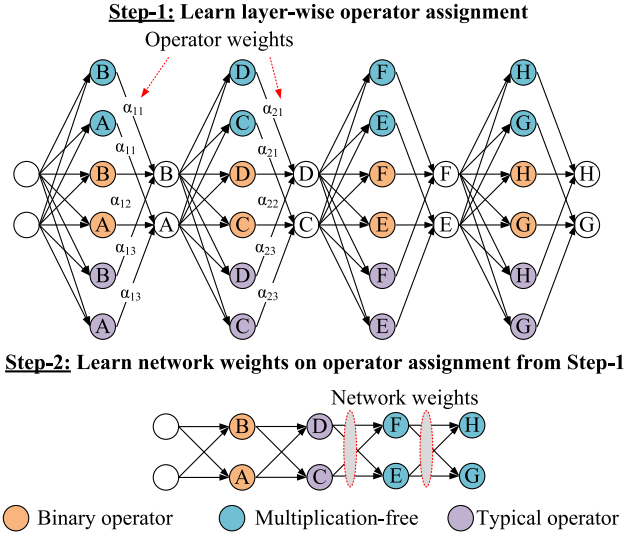

FIGURE 2. Learning architecture of Single and Bi-Level ENOS: In the first step, candidate operators' outputs are combined at each layer using linear interpolation with operator weights, α. The operator weights are also learned through back-propagation. Only the operators with the highest weight are retained across the network in the second step. The corresponding weights for the entire network are relearned. Since the cost function in both steps is continuous, typical learning tools such as ADAM can be utilized.

with the three chosen operators, typical (T), multiplication-free (MF), and binary connect (BC), are as following

$$f_T(\mathbf{x}, \mathbf{w}) = \sum x_i \cdot w_i, \tag{3a}$$

$$f_{MF}(\mathbf{x}, \mathbf{w}) = \sum \text{sign}(x_i) \cdot \text{abs}(w_i) + \text{sign}(w_i) \cdot \text{abs}(x_i), \tag{3b}$$

$$f_{BC}(\mathbf{x}, \mathbf{w}) = \sum x_i \cdot \text{binarize}(w_i). \tag{3c}$$

Here, $\cdot$ is an element-wise multiplication operator, $+$ is element-wise addition operator, and $\sum$ is vector sum operator. sign() operator is ± 1 and abs() operator produces absolute unsigned value.

Note that different operators considered in Eq. (3) are fundamentally different and are not just the precision truncated version of the typical operator in Eq. (3a). This makes our work different from automated mixed-precision learning [20]–[22]. Significantly, unlike simply constraining the precision of an operator, adapting the correlation function itself opens opportunities to *co-design* the underlying hardware, which can lead to dramatically lower operating energy and transmission overheads. E.g., in [8], utilizing multiplication-free operator in Eq. (3b), new computing modules such as memory-immersed data converters were presented. Prior works miss these opportunities in consideration by only considering layer-wise mixing of precision of a single operator. Meanwhile, developing a DNN automation framework to leverage such low-level hardware opportunities is one of this work's main focuses.

Although the binary operator in Eq. (3c) is non-differentiable, prior works [23]–[25] have presented approaches such as straight-through estimators to circumvent the problem. Similarly, sign() and abs() functions are non-differentiable in Eq. (3b). Prior work [26] approximates

Algorithm 1: Bi-Level ENOS (First-Order Approximation)

Goal: Extract trained θ^* and α^* to reduce the loss for Step-1. Loss function: $\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{acc}}(\theta, \alpha) + \lambda \sum_{i=1}^{N} N_{OP,i} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{OP,j} + \lambda \sum_{i=1}^{N} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{TX,ij}$
Given: Network weights: θ; Operator weights: α.
while *not converged do* **do**
  1. Update α by descending the loss $\Delta\mathcal{L}_{\text{net }Val}(\theta^t, \alpha)$
  2. Update θ descending the loss $\Delta\mathcal{L}_{\text{net }Train}(\theta, \alpha^t)$;
end
For Step-2, choose the operator with the highest weight factor (α_j) at each layer. Relearn the network weights.

the derivative of abs() with a steep hyperbolic tangent and the derivative of sign() with a steep Gaussian function to address this. Therefore, with the added modifications, the net loss function can still be optimized using typical gradient descent approaches. We experimented with two solvers, stochastic gradient descent (SGD) and ADAM, for our simulations. We found that while both solvers gave similar training and validation losses, ADAM was able to converge faster than SGD due to its adaptive learning rate and other hyperparameters. Therefore, we chose ADAM for all the ENOS approaches discussed in the paper. In the first step of a baseline approach to ENOS, following the above loss formulation, a candidate locally optimal parameter set θ^* and α^* can be extracted using gradient descent. Then, in the second step, a new design of DNN is considered where only the optimal operator based on the learned weight factors ($\max_j(\alpha_{ij})$) is considered at a layer i. The corresponding weights for the new DNN are then relearned.

In addition to this single-level baseline optimization approach to simultaneously extract optimal parameter set θ^* and α^*, we also consider a bi-level optimization procedure for step-1 of the learning procedure. Similar to [7], a bi-level optimization procedure is defined in our framework as

$$\min_{\alpha} \mathcal{L}_{\text{net }Val}(\theta^*(\alpha), \alpha) \tag{4a}$$

Such that;

$$\theta^*(\alpha) = \text{argmin}_{\theta} \mathcal{L}_{\text{net }Train}(\theta, \alpha) \tag{4b}$$

In the above, $\mathcal{L}_{\text{net }Val}$ is the loss function of Eq. (2) evaluated on the validation set whereas $\mathcal{L}_{\text{net }Train}$ is the loss function evaluated on the training set. Due to expensive inner optimization in Eq. (4), we follow the first-order approximation from [7] discussed in Algorithm 1, which trains the network iteratively on validation and training sets. On the top level, we update the weight factors α by descending the validation loss $\Delta\mathcal{L}_{\text{net }Val}$. Keeping the learned α, we descend $\Delta\mathcal{L}_{\text{net }Train}$ to learn the DNN weights θ. Likewise, sequentially, we iterate through the validation and training data set to learn the final α.

Algorithm 2: Sequential ENOS

Goal: Sequentially select the operator for network layers.
Loss function: $\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{acc}}(\theta, \alpha) + \lambda \sum_{i=1}^{N} N_{OP,i} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{OP,j} + \lambda \sum_{i=1}^{N} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{TX,ij}$
Given: Network weights: θ; Operator weights: α.
for *layer l=1 to L* **do**
  while *not converged* **do**
    Update α, θ by descending the loss on training set $\Delta\mathcal{L}_{\text{net }Train}(\theta, \alpha)$
  end
  1. Find the highest network weight factor (α_{ij}).
  2. Assign operator j to layer i and prevent further optimization of this choice.
end
Relearn the network weights.

B. SEQUENTIAL ENOS

In the previous approach, concurrent searching of operators for all layers is susceptible to overfitting. To address this concern, we also investigated a sequential mode of operator search. Under the sequential search, operator assignments are selected only for a single layer in one iteration while keeping the other layers' operator choices open. Specifically, a layer i is assigned the operator choice j if it has the highest weight factor α_{ij} among all layers and operator choices. All operator choices are considered in the following training iteration for the remaining unassigned network layers, and their corresponding operator weights are relearned. The iterations continue until all layer-wise operator assignments have been found. Algorithm 2 describes such sequential operator search procedures in ENOS.

C. STOCHASTIC ENOS

The optimal operator assignment problem in ENOS can also be treated under a stochastic framework. Specifically, $\text{softmax}(\alpha_{ij})$ can be seen as a multinomial distribution parameter representing the operator choices j on a layer i. Unlike the previous optimization procedures where a layer i is assigned the operator j if α_{ij} is the maximum, the assignments under the stochastic setting are made by sampling operator assignment based on the probability $\text{softmax}(\alpha_{ij})$. Here, a population of N candidate networks can be prepared by sampling operator assignments based on $\text{softmax}(\alpha_{ij})$. All candidate networks are then trained and evaluated. Subsequently, the best performing network from the candidate networks can be selected by testing it on the validation set. Algorithm 3 describes this procedure formally. In this algorithm, operator weights (α) and layer weights (θ) are learned for the entire network using bi-level optimization. After the training is complete, operator weights (α) are utilized to sample a population of networks (let's say total N networks). The sampled population where layer-wise operators have already been assigned based on α is

then retrained to determine the assigned operator's final layer weights (θ). Therefore, the final training step will be repeated for all networks in the population, i.e., N times.

Algorithm 3: Stochastic ENOS

Goal: Learn multinomial distribution parameters representing optimal operator choices and operator assignments.
Loss: $\mathcal{L}_{\text{net}} = \mathcal{L}_{\text{acc}}(\theta, \alpha) + \lambda \sum_{i=1}^{N} N_{OP,i} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{OP,j} + \lambda \sum_{i=1}^{N} \sum_{j=1}^{M} \text{softmax}(\alpha_{ij}) E_{TX,ij}$
Given: Network weights: θ; Operator weights: α.
Step-1: learn θ^* and α^*
while *not converged* **do**
  Update α, θ by descending the loss $\Delta \mathcal{L}_{\text{net}\,Train}(\theta, \alpha)$
end
Step-2: Create a population of candidate networks
for *samples n=1 to N* **do**
  for *layer l=1 to L* **do**
    Assign operator based on multinomial distribution parameters softmax(α_{ij}).
  end
  Relearn network weights
end
Step-3: Find the optimal operator assignment
for *samples n=1 to N* **do**
  Characterize the network on validation set.
end
Assign the most accurate network as the optimal choice.

III. ENOS-BASED DNN MAPPING WITH DIGITAL SPATIAL ACCELERATORS

This section discusses the potential of ENOS to accelerate the deep learning workload on digital spatial architecture. We present multiply-accumulate (MAC) cores in the architecture with reconfigurable cells to process various inference operators. ENOS exploits the architecture's reconfigurability to process various DNN layers in optimal core configuration to minimize net computing and data exchange energy. Through our evaluations of the digital architecture, we also compare the energy efficiency and energy scalability of various ENOS learning approaches.

A. SPATIAL DIGITAL ACCELERATOR WITH OPERATOR CONFIGURABLE MAC CELLS

Spatial digital accelerators have become prevalent for high-performance DNNs [27]–[29]. A spatial accelerator combines many parallel processing elements (PEs) and consists of a distributed memory hierarchy including input activation scratchpad, weight scratchpad, partial sum pad, and multiply-accumulate (MAC) units using a network-on-chip (NOC) similar to Figure 3(a). Memory organization for a DNN workload is optimally distributed among many parallel cores so that the inference flow can be processed with

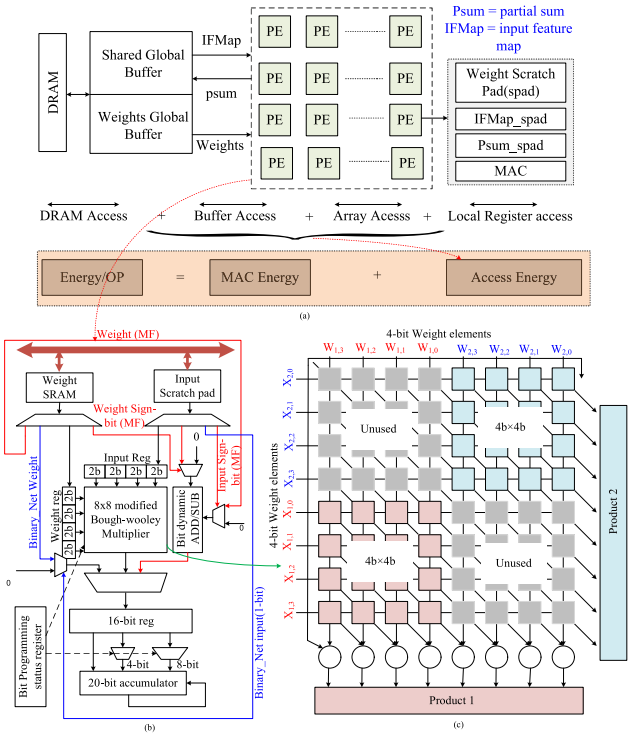

FIGURE 3. (a) Architecture of digital spatial accelerator evaluated under ENOS. External DRAM anchors to a shared global buffer for weights, input, and output feature map. The global buffer is connected to 168 processing engines (PE) using a bus where each PE is reconfigurable to various learning operators. Each PE houses scratchpad memory for weight, input feature maps, and product sum. (b) Digital MAC unit with reconfigurable operators. The datapaths for different operators are color-coded: datapaths for typical, multiplication-free & binary operators are highlighted with black, red & blue, respectively. (c) The multiplication in a typical operator is implemented using a reconfigurable 8-bit Bough-Wooley multiplier. At lower precision, the multiplier can perform parallel operations. For example, in the shown configuration, an 8-bit array-based multiplier performs two 4-bit multiplications in parallel.

minimal data movements and the least latency. Considering ENOS-based DNN mapping on a spatial accelerator, Figure 3 shows a digital core that can be reconfigured among the three operators we considered above. The MAC core also has precision reconfigurability for each operator. Thereby the optimal inference operator and corresponding optimal operating precision can be implemented flexibly at each layer.

In Figure 3(b), datapaths for different operators are associatively color-coded: datapaths for typical, multiplication-free & binary operators are highlighted with black, red, and blue, respectively. Although training for the network weights is performed considering a floating-point precision, a maximum of 8-bit fixed precision operation is considered during inference since, similar to prior work [30]–[32] where 8-bit fixed-precision inference suffers minimal accuracy degradation but can considerably simplify the hardware implementation and improve computations' energy efficiency. Datapath to implement a typical operator includes a modified 8-bit Baugh-Wooley multiplier to accommodate bit precision reconfigurability and a 20-bit adder (to accumulate the 16-bit outputs of the multiplier continually). The modified multiplier, built on an array-based circuit, reuses

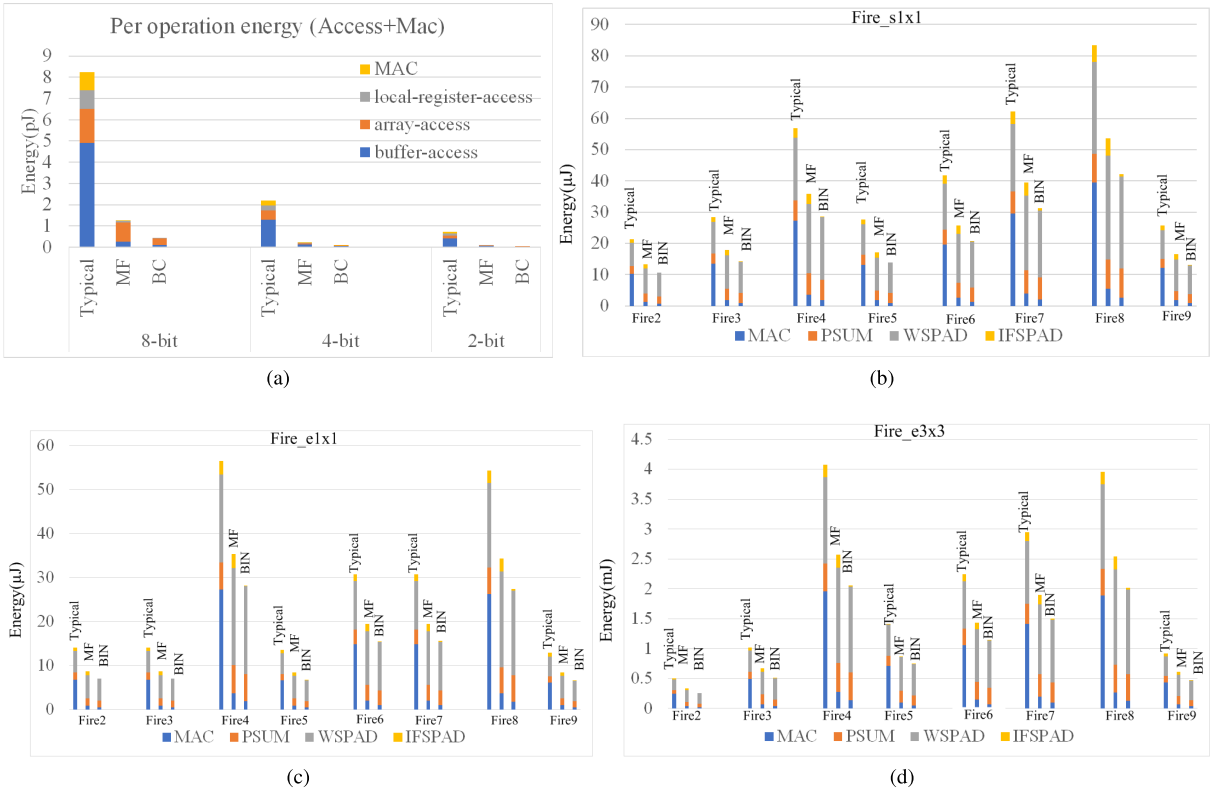

FIGURE 4. Energy distribution for SqueezeNet: (a) Per operation energy distribution at varying precision for typical, multiplication-free (MF) and binary connect (BC) operators. Each fire layer of SqueezeNet has two expansion blocks (*e*1 × 1 and *e*3 × 3) and one squeeze block (*s*1 × 1). (b)-(d) Layer wise energy distribution for *s*1 × 1, *e*1 × 1 and *e*3 × 3.

the inactive full adder cells during lower precision operation. This enables symmetric subword parallelism in lower precision, hence more energy savings than simply considering a lower precision of precision reconfigurable multiplier. This modified multiplier configuration enables $8/b$ independent multiplications in parallel with $b = 8, 4, 2$ bit precisions. Figure 3(c) shows the computation of *two* 4-bit multiplications simultaneously with the modified 8-bit Baugh-Wooley multiplier.

Meanwhile, the multiplication-free operator requires just an adder/subtractor block as it essentially implements addition/subtraction of input activation (I_{ACT}) depending on the sign of the corresponding weight parameter and addition/subtraction of weight depending on the sign of I_{ACT}. I_{ACT} and W can be configured to 8, 4, or 2-bits. Unlike typical and multiplication-free operators, weights in the binary operator are just one-bit while I_{ACT} is represented using 8, 4, or 2-bits. Hence, the binary operator adds/subtracts consecutive input activation depending on the corresponding weight bits and therefore needs only an adder/subtractor block. We use a 12-bit adder with multiplication-free and binary operators since they require a lower dynamic range of I_{ACT} than the typical operator. In the considered MAC unit, the adder's dynamic range can be configured to either 20-bit or 12-bit based on a switch.

The design of the above MAC unit was synthesized using Cadence RC Compiler. Predictive technology models (PTM) for 15nm CMOS technology [33] were used for the synthesis at nominal supply voltage. From the synthesis, we estimated the average energy of various operators for cost function terms $E_{OP,j}$ in Eq. (2). We used an Eyeriss-like architecture [27] of reconfigurable MAC cores with a total of 168 PEs for ENOS-based layer-wise optimal operator mapping. Timeloop [34] was used to find the optimal mapping of various layers. Accelergy [35] was used for pre-layout transmission energy estimation. The runtime action counts was modified according to the operator. Thereby, using the integrated methodology combining MAC core synthesis, Timeloop, and Accelergy, the average transmission energy of a layer i for operator j, $E_{TX,ij}$ was estimated for cost function in Eq. (2) on various benchmark DNN layers.

Figure 4(a) shows the MAC energy and the access energy comprised of buffer energy, PE-array access energy, and local register access energy for a unit operation for all three operators for 8, 4, and 2-bit precision. The typical operator is the most computationally expensive, whereas the binary operator requires ∼11.5× lower compute energy than the typical operator. Figure 4(b)-(d) show layer-wise energy consumption for 8-bit implementation of SqueezeNet. The figures show the detailed energy distribution of the fire layers, consisting of two expansion layers and one squeeze layer. Even though MAC energy is much lower in per-operation energy estimation than access energy, it is quite significant in overall layer-wise energy for the typical operator.

The expansion layer $e3 \times 3$ of SqueezeNet is much more computationally expensive than the $s1 \times 1$ and $e1 \times 1$. Within the expansion and squeeze layer, the energy follows a similar trend; fire4 and fire8 are the most energy-expensive layers for all $e3 \times 3$, $e1 \times 1$, and $s1 \times 1$ due to their greatest number of operations in the original SqueezeNet architecture. In the original SqueezeNet architecture, Fire-4 layer performs 46 million multiply-accumulate operations, and Fire-8 performs 59 million multiply-accumulate operations at each input. Even though ENOS adapts the operators of these layers to minimize their respective processing energy since these were the most expensive layers in the original architecture, they remain dominant even after the adaptation by ENOS.

B. SIMULATION RESULTS

We analyze the previous ENOS approaches on popular networks SqueezeNet [36] and ShuffleNet [37]. Note that unlike the predecessor DNNs such as VGG16 [38] and GoogleNet [39], SqueezeNet and ShuffleNet are already optimized for computational efficiency and edge computing, therefore are harder test-cases for further downscaling of their computing energy. The networks were trained on CIFAR10 and CIFAR100 to search the optimal operator assignments layer-wise. Figure 5 shows an exemplary synthesis of SqueezeNet using ENOS on CIFAR10 and CIFAR100 datasets. Rows in the table show optimal operator assignments under varying λ. Increasing λ allows trading-off accuracy against the DNN accelerator's operating energy. For CIFAR10, compared to a typical (T) operator on all layers, optimally integrating different operators on different layers allows $\sim$20% lower energy in ENOS at only $\sim$1.5% accuracy reduction. The lower rows in the figure show even more aggressive energy scaling while suffering moderate accuracy degradation.

The application of ENOS on the network also offers many interesting insights. With increasing λ to improve energy efficiency, operator assignments change only in the middle convolution layers from typical to multiplication-free and then binary. Meanwhile, operator assignments for the initial and last convolution layers remain typical. This characterization illustrates that feature extraction in the middle layers in the network is more amenable to simplification and accelerator energy saving without a considerable effect on the overall network's output accuracy.

Figure 6(a) shows the evolution of operator assignments on SqueezeNet under sequential ENOS. We kept the operator for the first convolution layer (Conv1) as typical (T) since, under bi-level ENOS, the layer preferred standard operator even under aggressive λ scaling. Among the remaining Fire layers, Fire7 is first assigned binary (B) operator since it showed the highest propensity towards B operating under accelerator energy scaling constraints among all the other layers. The operator choices remained open for the different layers. Subsequently, Fire6 is assigned B operator, and Fire1 is given T operator at the last cycle. Compared to single-level and bi-level ENOS, the training workload for sequential ENOS is substantially more. Note that the number of training iterations in sequential ENOS scale proportionally to the number of network layers.

Figure 6(b) shows a sample population of operator assignments on various candidate networks under stochastic ENOS. We extended stochastic ENOS to assign operating precision and inference operator for this result. To extend ENOS for both operator and precision choices, the output of each layer is defined as $\overline{y_i} = \sum_{j=1}^{N} \sum_{k=1}^{K} \text{softmax}(\alpha_{ijk}) y_{ijk}(x_i)$, where K is the total number of precision options considered for each operator j for layer i. $y_{ijk}(x_i)$ is the output at layer i from operator j at precision k. Weight factors α_{ijk} are learned using the procedure in Algorithm 3, and $\text{softmax}(\alpha_{ijk})$ are treated as parameters of a multi-nominal distribution from where candidate networks are sampled. Interestingly, DNN layers show varying degrees of entropy in selecting the optimal layer operator and precision. For example, in the shown table, layers Fire2 and Fire7 show less entropy and mostly opt for T and MF operators, respectively. Meanwhile, layers such as Fire4 and Fire5 show much higher entropy in their operator assignment and choose from a larger operator and precision set in various sampling iterations. Notably, such layer-wise operator assignments among various candidate networks are also jointly correlated as the network searches for the best operator combinations to maximize accuracy under accelerator energy constraints.

C. COMPARISON OF ENOS APPROACHES

Prior mixed-precision training frameworks [20]–[22] optimally assign precision to each layer while considering only the typical inference operator (Eq. 3(a)) throughout the network. Comparably, the proposed framework adds greater flexibility to energy-constrained learning by allowing each layer to choose among a set of network operators and operating precision. As we mentioned earlier, simplifying the learning operator can lead to significant energy saving, e.g., multiplication-free operators in Eq. 3(b) obviate energy-expensive multiplier modules. Figure 7 characterizes the advantage of our 'mixed-operator + mixed-precision' approach over 'mixed-operator' and 'mixed-precision' learning. SqueezeNet and ShuffleNet are characterized on CIFAR10 and CIFAR100 under the three settings. For 'mixed-precision,' only the typical operator with optimally chosen input and weight precision among 2-bit, 4-bit, and 8-bit is selected at each layer. For 'mixed-operator,' an operator among typical, multiplication-free, and binary-connect is chosen at each layer while processing at 8-bit precision. Meanwhile, the 'mixed-operator + mixed-precision' approach has the flexibility to choose the optimal operator *and* operating precision at each layer. Under all three settings, the bi-level optimization procedure is followed. Different DNN configurations are estimated based on the operator energies in Figure 4(a). As discussed earlier, the transmission energy of operator-layer tuples is extracted from Accelergy on the accelerator configuration in Figure 3. The figure shows normalized energies of different configurations

SqueezeNet: Cifar10

	Accuracy (%)	Energy	Conv1	Fire2			Fire3			Fire4			Fire5			Fire6			Fire7			Fire8			Fire9		
				S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1
Uni-Operator	91.6	1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	77	0.34x	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF
	61	0.17x	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
ENOS	89.3	0.76x	T	T	T	T	T	MF	T	T	MF	MF	T	MF	T	B	MF	B	T	T	T	T	T	MF	T	T	T
	88.25	0.65x	T	T	MF	T	T	MF	T	T	MF	B	B	MF	B	T	MF	MF	T	MF	MF	B	T	MF	MF	MF	T
	85.12	0.43x	T	MF	MF	MF	T	MF	MF	B	B	B	T	MF	MF	MF	T	B	B	MF	T	T	MF	MF	MF	T	T

SqueezeNet: Cifar100

	Accuracy (%)	Energy	Conv1	Fire2			Fire3			Fire4			Fire5			Fire6			Fire7			Fire8			Fire9		
				S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1
Uni-Operator	69.7	1	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
	56.1	0.34x	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF	MF
	49	0.17x	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B	B
ENOS	67.8	0.81x	T	T	T	T	T	MF	T	T	T	T	T	MF	T	B	MF	B	T	T	T	T	T	MF	T	T	T
	64.15	0.69x	T	T	MF	T	T	MF	T	T	MF	B	B	MF	MF	T	MF	MF	T	MF	MF	MF	T	MF	MF	MF	T
	63.9	0.46x	T	MF	MF	MF	T	MF	MF	B	B	B	T	MF	MF	MF	T	MF	MF	MF	T	T	MF	MF	MF	T	T

FIGURE 5. Layer-wise operator mapping among typical (T), multiplication-free (MF), and binary connect (B) for SqueezeNet network for CIFAR10 and CIFAR100 data set. The hyperparameter λ in the cost function controls the accuracy-energy trade-off. By reducing λ, the network is forced to operate at lower energy while maintaining the highest accuracy possible.

Sequential selection of operators for SqueezeNet

	Conv1	Fire1	Fire2	Fire3	Fire4	Fire5	Fire6	Fire7	Fire8
Cycle 1	T							B	
Cycle 2				Undecided Layers			B		
Cycle 3						B			
Cycle 4									B
Cycle 5					MF				
Cycle 6				B					
Cycle 7			MF						
Cycle 8		T							

(a)

	Conv1	Fire2			Fire3			Fire4			Fire5			Fire6			Fire7			Fire8			Fire9		
		S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1	S1	E3	E1
Sample1	T	T	T	T	T	T	T	T	MF	T	B	B	T	T	MF	MF	T	B	T	B	MF	T	T	T	T
Sample2	T	T	T	T	MF	MF	MF	MF	T	T	MF	MF	MF	MF	MF	MF	MF	T	T	T	MF	MF	MF	MF	T
Sample3	T	B	T	T	T	B	B	B	T	T	T	T	T	T	B	B	B	MF	MF	MF	B	B	B	B	T
Sample4	T	T	T	T	T	MF	T	T	MF	MF	T	MF	T	B	T	T	T	T	T	T	T	MF	T	T	T
Sample5	T	T	MF	T	T	MF	T	T	MF	B	B	MF	B	T	MF	MF	T	MF	MF	B	T	MF	MF	MF	T
Sample6	T	MF	MF	MF	T	MF	MF	B	B	B	T	MF	MF	MF	T	B	T	T	T	T	MF	MF	MF	T	T

MF-8bit MF-4bit MF-2bit Typical-8bit Typical-4bit Typical-2bit BinaryConnect-8bit BinaryConnect-4bit BinaryConnect-2bit

(b)

FIGURE 6. (a) Evolution of operator assignments on the layers of SqueezeNet under sequential optimization in ENOS. Only one layer is assigned an operator in one time-step. (b) The operator assignments on various candidate networks under stochastic ENOS by considering both optimal precision and optimal operator assignment problem at each layer.

under the baseline case where all operations are done using the typical (T) operator. From Figure 7, 'mixed-operator + mixed-precision' offers ∼2-5% energy improvement over mixed-operator and ∼5-7% energy improvement over mixed-precision implementation under same accuracy constraints. An interesting observation is that mixed-operator implementation is more accurate for lower energy budgets, whereas mixed-precision performs better for higher energy budgets.

Figure 8 compares different optimization procedures – single-level, bi-level, and sequential ENOS – on the same networks and datasets. In the figure, bi-level ENOS outperforms single-level ENOS under the same energy constraints. For single-level optimization, learning network weights θ and operator weights α on the same dataset, i.e., training set, makes the training vulnerable to over-fitting. Meanwhile, bi-level optimization learns θ on the training set and α on the validation set, improving learning by considering more data. Especially, bi-level optimization shows significant improvement in accuracy on more challenging datasets, i.e., on CIFAR100 compared to CIFAR10. Figure 8 also compares single-level and bi-level ENOS against sequential ENOS. Sequential ENOS significantly improves the operator assignments compared to the concurrent operator assignment modes (single and bi-level). For example, for SqueezeNet on CIFAR100 [Figure 8(a)], sequential ENOS results in a network configuration that has >10% accuracy than single and bi-level ENOS when the energy budget is only 40% of the baseline all typical operator case. Single and bi-level ENOS modes consider significant accuracy degradation; sequential ENOS incurs only ∼5% accuracy loss when the accelerator energy is constrained to be one-fifth of the maximum considered here. Similarly, for ShuffleNet on CIFAR100 [Figure 8(b)], sequential ENOS suffers a minimal degradation in energy even when the network energy is scaled down to about 30% of the baseline case. Here, single and bi-level

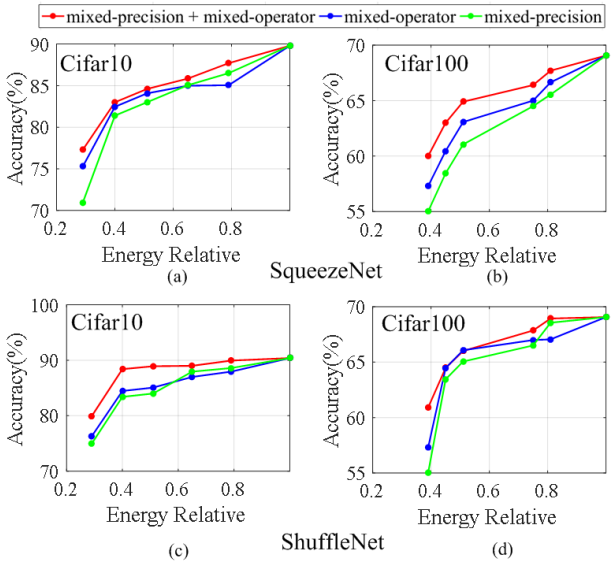

FIGURE 7. Comparison of Energy vs. Accuracy on CIFAR10 and CIFAR100 for mixed-precision, mixed-operator, and 'mixed-precision + mixed-operator' approaches: (a) SqueezeNet on CIFAR10, (b) SqueezeNet on CIFAR100, (c) ShuffleNet on CIFAR10, and (d) ShuffleNet on CIFAR100.

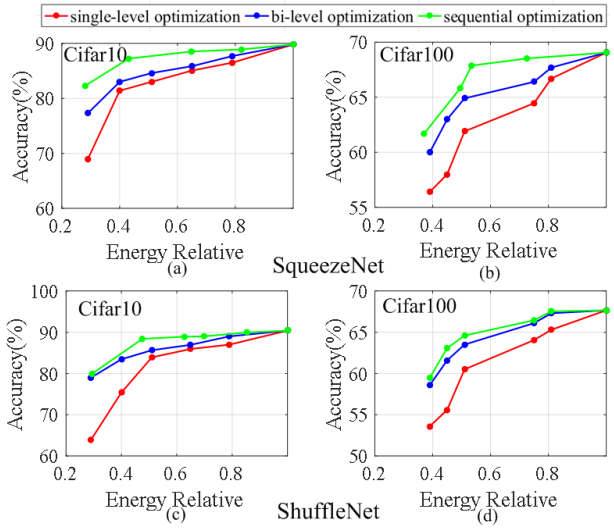

FIGURE 8. Comparison of Energy vs. Accuracy on CIFAR10 and CIFAR100 for single-level optimization, bi-level optimization, and sequential optimization under 'mixed-precision + mixed-operator' learning: (a) SqueezeNet on CIFAR10, (b) SqueezeNet on CIFAR100, (c) ShuffleNet on CIFAR10, and (d) ShuffleNet on CIFAR100.

modes of ENOS cannot scale down the network energy to such lower levels. Nonetheless, sequential ENOS is much more computationally expensive than single and bi-level modes; under sequential ENOS, the training iterations grow in sequential ENOS as the number of layers increases.

Figure 10 shows stochastic implementation's accuracy vs. energy performance. The figure shows that the sampled network population has varying energy-accuracy trade-offs. Under extreme energy constraints, stochastic ENOS degrades more gracefully and provides better accuracy in the synthesized networks than sequential ENOS. For example, comparing Figures 8 and 9, even when the operating energy reduction

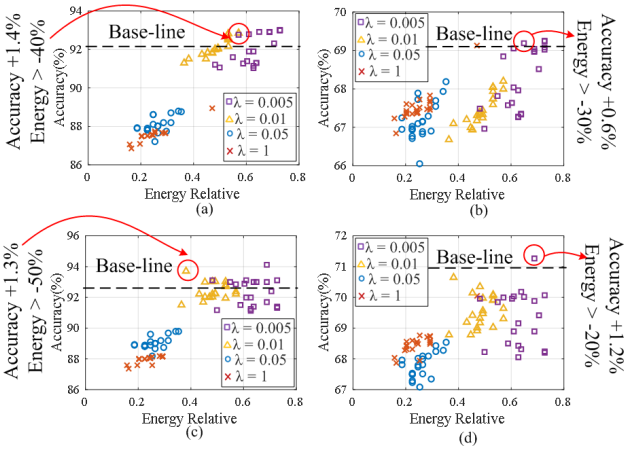

FIGURE 9. Energy vs. Accuracy on CIFAR10 and CIFAR100 for (a)-(b) SqueezeNet and (c)-(d) ShuffleNet under stochastic mode of ENOS. a higher λ forces the network to minimize its energy more aggressively at the cost of lower accuracy. In the figure, each symbol corresponds to a value of λ in equation 2a (loss function). Many points of the same symbol type and color on accuracy – energy axes show the characterization of networks generated by the same λ.

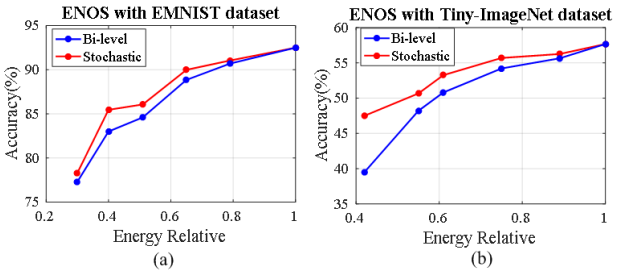

FIGURE 10. Evaluation of ENOS on additional datasets: (a) EMNIST and (b) Tiny ImageNet. Similar to the considered cases in the original manuscript, ENOS is able to trade-off energy with limited loss in accuracy for EMNIST. Stochastic ENOS loses 1.9% accuracy at 30% energy gain. Stochastic mode of ENOS outperforms bi-level optimization in both cases.

is ∼20%, the stochastic ENOS maintains a graceful degradation of accuracy of ∼2%, whereas it is ∼10% for sequential ENOS. Compared to sequential ENOS, stochastic ENOS is also more computationally efficient. In sequential ENOS, the number of training iterations grows with the number of deep learning layers since each iteration determines the operator for one layer. In stochastic ENOS, meanwhile, the entire network is trained only once. However, this is also true that in stochastic ENOS the final training step on determining final layer weights needs to be done as many times as the sampled population size. Meanwhile, the final weight training step in sequential ENOS is performed only once. Still, the combined training of θ and α is much more expensive than training the final layer weights. Therefore, sequential ENOS was found to be more costly than stochastic ENOS.

We further validate the ENOS approach by extending the application of ENOS on Extended-MNIST (E-MNIST) and Tiny ImageNet datasets. These datasets are sufficiently complex and limited in size to help us perform rigorous simulations within limited computing resources/time. This time we choose a non-optimal network like GoogleNet [39] instead of an already optimized network for computation

like SqueezeNet or ShuffleNet. GoogleNet is a 22-layer deep convolutional neural network that is a variant of the Inception Network, a Deep Convolutional Neural Network developed by researchers at Google. Figure 10(a) discusses the accuracy-vs-energy results for E-MNIST. EMNIST is an MNIST-like dataset consisting of various letters and digits. EMNIST-Balanced consists of 47 classes containing a total of 131600 samples. ENOS can minimize 30-40% energy while incurring only 2-3% accuracy loss. Similar to other datasets, the stochastic mode of ENOS shows better accuracy than bi-level optimization. Figure 10(b) shows similar results for tiny-ImageNet. Tiny ImageNet is a subset of the famous ImageNet Dataset, which contains 100,000 images in 200 classes. The images are downsized to 64 × 64-colored images compared to original images of size 224 × 224 images.

IV. CONCLUSION

This paper presented a novel class of efficient energy-aware operator search algorithms (ENOS) for various deep neural networks. ENOS can match the state-of-the-art performance metrics for neural networks on image classification with remarkable energy efficiency improvement. It can be applied to other domains/applications, such as Transformer-based language models. Additionally, we present a reconfigurable MAC core that implements the chosen optimal operators by carefully considering the energy-accuracy trade-offs. In the future, the layer-level operator search approach presented here can be expanded to neuron-level operator search. The ENOS approach can also be widened to incorporate similar goals in multiple resource-constrained settings beyond energy efficiency, such as latency constraints.

REFERENCES

[1] R. Huang, J. Pedoeem, and C. Chen, "YOLO-LITE: A real-time object detection algorithm optimized for non-GPU computers," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 2503–2510.

[2] A. Shylendra, P. Shukla, S. Mukhopadhyay, S. Bhunia, and A. R. Trivedi, "Low power unsupervised anomaly detection by nonparametric modeling of sensor statistics," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 28, no. 8, pp. 1833–1843, Aug. 2020.

[3] A. Shylendra, P. Shukla, S. Bhunia, and A. R. Trivedi, "Analog-domain time-series moment extraction for low power predictive maitenance analytics," in *Proc. IEEE Int. Conf. Artif. Intell. Circuits Syst. (ISCAS)*.

[4] P. Shukla, A. Muralidhar, N. Iliev, T. Tulabandhula, S. B. Fuller, and A. R. Trivedi, "Ultralow-power localization of insect-scale drones: Interplay of probabilistic filtering and compute-in-memory," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 1, pp. 68–80, Jan. 2021.

[5] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," 2016, *arXiv:1611.01578*.

[6] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," 2018, *arXiv:1802.03268*.

[7] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," 2018, *arXiv:1806.09055*.

[8] S. Nasrin, D. Badawi, A. E. Cetin, W. Gomes, and A. R. Trivedi, "MF-Net: Compute-in-memory SRAM for multibit precision inference using memory-immersed data conversion and multiplication-free operators," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 5, pp. 1966–1978, May 2021.

[9] S. Nasrin, P. Shukla, S. Jaisimha, and A. R. Trivedi, "Compute-in-memory upside down: A learning operator co-design perspective for scalability," in *Proc. IEEE Design Autom. Test Eur. (DATE)*, Feb. 2021, pp. 890–895.

[10] M. Elhoushi, Z. Chen, F. Shafiq, Y. H. Tian, and J. Y. Li, "DeepShift: Towards multiplication-less neural networks," 2019, *arXiv:1905.13298*.

[11] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarized neural networks: Training deep neural networks with weights and activations constrained to +1 or −1," 2016, *arXiv:1602.02830*.

[12] M. Courbariaux, Y. Bengio, and J. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," 2015, *arxiv:1511.00363*.

[13] S. Nasrin, S. Ramakrishna, T. Tulabandhula, and A. R. Trivedi, "Supported-BinaryNet: Bitcell array-based weight supports for dynamic accuracy-energy trade-offs in SRAM-based binarized neural network," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[14] P. Shukla, A. Shylendra, T. Tulabandhula, and A. R. Trivedi, "MC2RAM: Markov chain Monte Carlo sampling in SRAM for fast Bayesian inference," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[15] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[16] T. Simons and D.-J. Lee, "A review of binarized neural networks," *Electronics*, vol. 8, no. 6, p. 661, Jun. 2019.

[17] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: ImageNet classification using binary convolutional neural networks," in *Proc. Eur. Conf. Comput. Vis.* Springer, 2016, pp. 525–542.

[18] Y. LeCun, C. Cortes, and C. Burges, "MNIST handwritten digit database," AT&T Labs, Tech. Rep., 2010, vol. 2. [Online]. Available: http://yann.lecun.com/exdb/mnist

[19] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[20] P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh, and H. Wu, "Mixed precision training," 2017, *arXiv:1710.03740*.

[21] X. Jia, S. Song, W. He, Y. Wang, H. Rong, F. Zhou, L. Xie, Z. Guo, Y. Yang, L. Yu, T. Chen, G. Hu, S. Shi, and X. Chu, "Highly scalable deep learning training system with mixed-precision: Training ImageNet in four minutes," 2018, *arXiv:1807.11205*.

[22] D. Das, N. Mellempudi, D. Mudigere, D. Kalamkar, S. Avancha, K. Banerjee, S. Sridharan, K. Vaidyanathan, B. Kaul, E. Georganas, A. Heinecke, P. Dubey, J. Corbal, N. Shustrov, R. Dubtsov, E. Fomenko, and V. Pirogov, "Mixed precision training of convolutional neural networks using integer operations," 2018, *arXiv:1802.00930*.

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

[24] H. Yang, M. Fritzsche, C. Bartz, and C. Meinel, "BMXNet: An open-source binary neural network implementation based on MXNet," in *Proc. 25th ACM Int. Conf. Multimedia*, 2017, pp. 1209–1212.

[25] J. W. T. Peters and M. Welling, "Probabilistic binary neural networks," 2018, *arXiv:1809.03368*.

[26] C. E. Akbaş, A. Bozkurt, A. E. Çetin, R. Çetin-Atalay, and A. Üner, "Multiplication-free neural networks," in *Proc. 23rd Signal Process. Commun. Appl. Conf. (SIU)*, May 2015, pp. 2416–2418.

[27] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE J. Solid-State Circuits*, vol. 52, no. 1, pp. 127–138, Nov. 2017.

[28] N. Iliev and A. R. Trivedi, "Low latency CMOS hardware acceleration for fully connected layers in deep neural networks," 2020, *arXiv:2011.12839*.

[29] B. Kim, S. Lee, A. R. Trivedi, and W. J. Song, "Energy-efficient acceleration of deep neural networks on realtime-constrained embedded edge devices," *IEEE Access*, vol. 8, pp. 216259–216270, 2020.

[30] J. Su *et al.*, "A 28 nm 64 kb inference-training two-way transpose multibit 6T SRAM compute-inmemory macro for AI edge chips," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 240–242.

[31] J. Yue, Z. Yuan, X. Feng, Y. He, Z. Zhang, X. Si, R. Liu, M.-F. Chang, X. Li, H. Yang, and Y. Liu, "A 65 nm computing-in-memory-based CNN processor with 2.9-to-35.8TOPS/W system energy efficiency using dynamic-sparsity performance-scaling architecture and energy-efficient inter/intra-macro data reuse," in *IEEE Int. Solid-State Circuits Conf. (ISSCC) Dig. Tech. Papers*, Feb. 2020, pp. 234–236.

[32] A. Sayal, S. Fathima, S. T. Nibhanupudi, and J. P. Kulkarni, "COMPAC: Compressed time-domain, pooling-aware convolution CNN engine with reduced data movement for energy-efficient AI computing," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 2205–2220, Jul. 2021.

[33] S. Sinha, G. Yeric, V. Chandra, B. Cline, and Y. Cao, "Exploring sub-20 nm FinFET design with predictive technology models," in *Proc. 49th Annu. Design Autom. Conf. (DAC)*, 2012, pp. 283–288.
[34] A. Parashar, P. Raina, Y. S. Shao, Y.-H. Chen, V. A. Ying, A. Mukkara, R. Venkatesan, B. Khailany, S. W. Keckler, and J. Emer, "Timeloop: A systematic approach to DNN accelerator evaluation," in *Proc. IEEE Int. Symp. Perform. Anal. Syst. Softw. (ISPASS)*, Mar. 2019, pp. 304–315.
[35] Y. N. Wu, J. S. Emer, and V. Sze, "Accelergy: An architecture-level energy estimation methodology for accelerator designs," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Design (ICCAD)*, Nov. 2019, pp. 1–8.
[36] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, "SqueezeNet: Alexnet-level accuracy with 50X fewer parameters and <0.5 MB model size," 2016, *arXiv:1602.07360*.
[37] X. Zhang, X. Zhou, M. Lin, and J. Sun, "ShuffleNet: An extremely efficient convolutional neural network for mobile devices," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2017, pp. 6848–6856.
[38] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.
[39] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 1–9.

SHAMMA NASRIN (Graduate Student Member, IEEE) received the B.S. degree in electronics engineering from the Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2015. She is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Illinois at Chicago, IL, USA. Her research interests include neuromorphic computing, deep learning neural networks, and low-power and high-performance in-memory processing systems.

AHISH SHYLENDRA (Member, IEEE) received the B.E. degree in electronics and communication engineering from the Global Academy of Technology, Bengaluru, in 2014, the M.Tech. degree in VLSI from the National Institute of Technology Goa, Ponda, India, in 2016, and the Ph.D. degree in electrical engineering and computer science from the University of Illinois at Chicago, in 2022. His current research interests include hardware security, emerging devices, analog, and mixed-signal design.

NASTARAN DARABI (Graduate Student Member, IEEE) received the B.Sc. degree in electrical engineering with a specialization in systems and control from the Sharif University of Technology. She joined the AEON Laboratory as a Ph.D. Student, in Spring 2021. Her current research interests include on-chip learning schemes, such as reinforcement learning and quantum machine learning, also efficient hardware-software co-operated systems for probabilistic AI, and nano-robotics.

THEJA TULABANDHULA received the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, USA, in 2014. He is currently an Assistant Professor with the Department of Information and Decision Sciences, University of Illinois at Chicago. His research interests include developing new methods for reinforcement learning, applying deep learning solutions to multiple application areas, and developing new optimization techniques.

WILFRED GOMES received the Bachelor of Technology degree in electronic engineering from the National Institute of Technology, Calicut, in 1992, and the master's degree in electrical engineering from the University of Hawaii, in 1995. From 1994 to 1997, he worked at Cadence Design on RTL and logic synthesis. He joined Intel, in 1997. He is currently a Principal Engineer. He has worked on the synthesis of domino control logic and the development of Register File synthesis tools. He has also been involved in co-optimizing process, design, and micro-architecture to achieve several generations of Intel microprocessors' frequency, area, and power goals.

ANKUSH CHAKRABARTY (Senior Member, IEEE) received the Ph.D. degree from Purdue University, West Lafayette, IN, USA, for developing scalable, data-driven methods for simplifying computationally intensive operations encountered in controlling and observing complex, nonlinear systems. He is currently a Principal Research Scientist with the Mitsubishi Electric Research Laboratory (MERL), Cambridge, MA, USA. He was a Ross Fellow with Purdue University, West Lafayette. Before joining MERL, he was a Postdoctoral Fellow at Harvard University, Cambridge, MA, USA, where he designed embedded predictive model controllers and deep learning-assisted control strategies for treating people with type one diabetes. His research interests include machine learning for control, optimization, and estimation, with application to sustainable building energy systems.

AMIT RANJAN TRIVEDI (Senior Member, IEEE) received the B.Tech. and M.Tech. degrees in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 2008, and the Ph.D. degree in electrical and computer engineering from the Georgia Institute of Technology, Atlanta, GA, USA, in 2015. He was with the IBM Semiconductor Research and Development Center (SRDC), Bengaluru, India, from 2008 to 2010. During his Ph.D. degree, he was a Summer Intern at IBM T. J. Watson Research Institute, in 2012, and Intel's Circuits Research Laboratory, in 2014. Since October 2015, he has been with the Department of Electrical and Computer Engineering, University of Illinois at Chicago, IL, USA, where he is currently an Assistant Professor. He has authored or coauthored over 70 papers in refereed journals and conferences. His current research interests include machine learning at the edge, hardware security, and energy-efficient systems. He received the IEEE Electron Device Society (EDS) fellowship, in 2014, where he was one of the three recipients worldwide. He also received the Georgia Tech Sigma Xi best Ph.D. dissertation Award. In 2021, he was also awarded the NSF CAREER Award.