Synthesizing Zero-loss Low-Power Approximate DNN Accelerators with Large-Scale Search

***Abstract*—Approximate computing is a promising way to improve the power efficiency of deep learning. While recent work proposes new arithmetic circuits (adders and multipliers) that consume substantially less power at the cost of computation errors, these approximate circuits decrease the end-to-end accuracy of common models. We propose AutoApprox, a framework to automatically generate approximate low-power deep learning accelerators without any accuracy loss. AutoApprox generates a wide range of approximate ASIC accelerators with a TPUv3 systolic-array template. AutoApprox uses a learned router to dynamically assign each DNN layer to an approximate systolic array from a bank of arrays with varying approximation levels. By tailoring this routing for a specific neural network architecture, we discover circuit designs without the dramatic accuracy penalty from prior methods. Moreover, AutoApprox optimizes for the end-to-end performance, power and area of the the whole chip and PE mapping rather than simply measuring the performance of the arithmetic units in isolation. To our knowledge, our work is the first to demonstrate the effectiveness of custom-tailored approximate circuits in delivering significant chip-level energy savings with zero accuracy loss on a large-scale dataset such as ImageNet. AutoApprox synthesizes a novel approximate accelerator based on the TPU that reduces end-to-end power consumption by 3.2% and area by 5.2% at a sub-10nm process with no degradation in ImageNet validation top-1 and top-5 accuracy.**

***Index Terms*—approximate computing, deep learning**

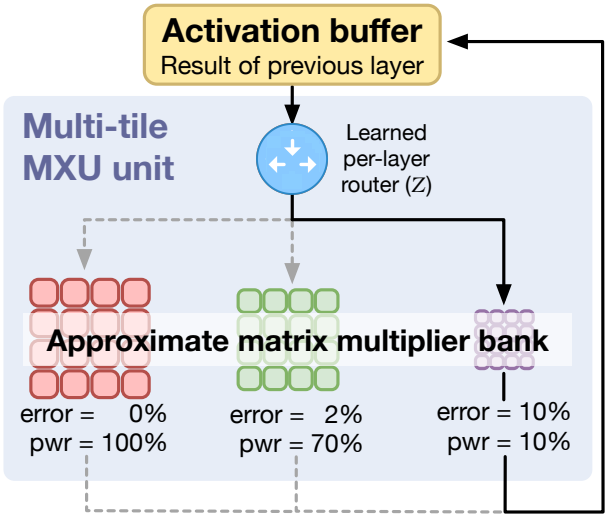

Fig. 1. AutoApprox is a full-stack framework to synthesize approximate systolic-array accelerators without accuracy loss. We achieve power savings without loss by co-designing the approximate units on a chip with the software mapping of layers to systolic arrays. At runtime, this enables dynamic routing of error-tolerant layers to more approximate cores, thereby yielding significant power savings.

I. Introduction

While the continued scaling of neural networks has enabled higher task accuracy, large models are increasingly energy-intensive to deploy. For example, model serving constitutes the majority (up to 80-90%) of deep learning workloads at Facebook and Amazon AWS [16, 41]. Even small efficiency improvements in inference accelerators will therefore greatly reduce the global energy consumption of deep learning.

In systolic-array accelerators, Zimmer et al. [46] report that over 30% of PE energy is consumed by arithmetic units that perform basic mathematical operations [38]. The current practice to improve the power-efficiency of these arithmetic units is to substitute full-precision floating-point calculations with low-bit precision quantized operations such as 8-bit arithmetic [13]. However, low-bit quantization comes at the cost of degraded accuracy [6]. In fact, the optimal precision for an architecture varies widely between layers [8].

Emerging work proposes novel approximate circuits that are dramatically more power-efficient than quantized operators [2]. These approximate operators (multipliers and adders) do not simply reduce the bit-precision of exact arithmetic but rather *tailor approximations to a specific numerical distribution* observed deployment. Approximate circuits thereby enable a better power-accuracy trade-off than quantization which uniformly approximates all inputs. In Figure 2, we visualize the error for a single approximate multiplier we study. This multiplier concentrates error on select sparse values but consumes $\mathbf{3.61}\times$ less energy than an 8-bit exact multiplier.

Prior work in approximate computing for DNNs finds accuracy drops under errors [43, 33]. Even if accuracy remains high, these approaches test small-scale models and datasets. For example, Mrazek et al. [34] approximates just one layer of an 8-layer ResNet.

How to utilize approximate cores while preserving high end-task accuracy? We take advantage of dark silicon [15] by instantiating additional approximate systolic arrays adjacent to an optional exact systolic array, as shown in Figure 1. At runtime, we dynamically route error-tolerant layers of a DNN to an approximate array with low dynamic power consumption. More sensitive layers are evaluated on the exact systolic array.

We propose AutoApprox, a framework to **automatically synthesize low-power approximate ASIC accelerators with zero accuracy loss** with no retraining required. Using an modified Edge TPU template, AutoApprox generates a diverse set of efficient designs with a reconfigurable routing array to a bank of systolic arrays containing open-source

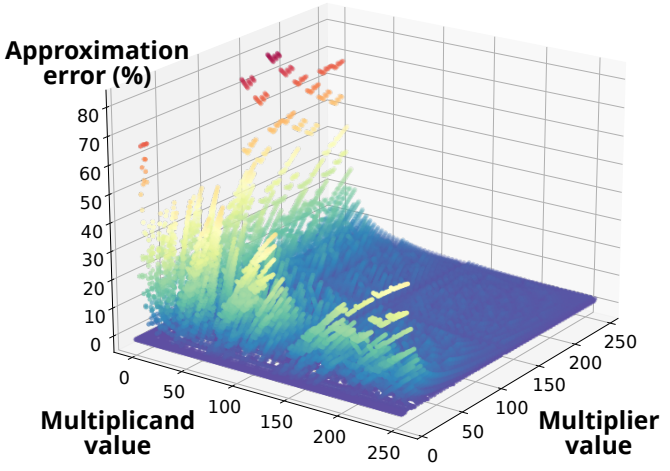

Fig. 2. Approximate multipliers trade-off exact computation in order to improve power-efficiency. The visualized multiplier consumes 3.61x less energy than a quantized multipler at the cost of 4.2% relative error. By carefully matching approximate multipliers with each layer in a neural network, we are able to reduce inference energy usage with zero loss in end-to-end accuracy.

approximate multipliers [32]. By co-optimizing the mapping of approximate systolic arrays to layers, we avoid accuracy losses due to homogeneous approximation [8] by custom-tailoring the approximation for different fine-grained portions of the computation graph. We also perform proof-of-concept evaluations on large-scale datasets (including ImageNet) and models, a first in the domain of functionally-approximate circuits for deep learning.

We evaluate AutoApprox on the ResNet-50 architecture [18] trained on the ImageNet dataset [5]. On a sub-10nm process node, we demonstrate our search methodology is Pareto-optimal when compared to other baselines. AutoApprox realizes zero ImageNet validation accuracy loss for ResNet-50 with savings of up to 3.2% in power consumption and up to 5.2% in area.

We make the following novel contributions:

- We propose AutoApprox, a framework for the design of zero-loss approximate DNN accelerators.
- We evaluate the accuracy of approximate designs with end-to-end simulation on significantly larger datasets than prior work. In order to make end-to-end evaluation tractable, we accelerate circuit simulation 7200× using caching and matrix decomposition.
- In order to search the combinatorial space of accelerator designs (up to $O(2^{268})$), we develop a Bayesian optimization solver to efficiently find high-quality solutions. We demonstrate this framework outperforms competitive baseline methods.
- We demonstrate AutoApprox improves the power-efficiency of an TPU-based design without any added area while maintaining end-to-end accuracy. For the first time to our knowledge, we evaluate functionally approximate DNN accelerators on a large-scale dataset and workload with no accuracy loss.

TABLE I
OVERVIEW OF PRIOR APPROXIMATE COMPUTING METHODS FOR DEEP NETWORKS WITH COMPARISION OF KEY FEATURES

	Largest dataset	Model MACs	Retrain free?	Zero loss?
Venkataramani et al. [43]	CIFAR-10	<1M	✗	✗
Zhang et al. [45]	CALTECH	<1M	✗	✗
Sarwar et al. [37]	CIFAR-100	<1M	✗	✗
Mrazek et al. [34]	CIFAR-10	21M	✓	✗
Mrazek et al. [33]	CIFAR-10	120M	✓	✗
AutoApprox (ours)	ImageNet-1k	2B	✓	✓

II. Related Work

Approximate computing: The slowdown of Moore's law and Dennard Scaling forces new deep learning accelerators to explore new ways to improve power-efficiency through specialization. Approximate computing promises a new era of custom approximate circuits with significant improvements in performance at the cost of degraded quality. Analog computing [3] offers large potential power savings; however, these technologies remain difficult to test and deploy as they are non-deterministic and result in a large accuracy degradation. We instead focus on *functionally-approximate circuits* which unlock power savings by replacing power-intensive segments of a circuit with inaccurate but simpler components. As an example, prior work Kim et al. [25] has shown that removing the carry and overflow logic from a 16-bit adder can yield a 2.3x more energy-efficient design. There are a variety of these manually designed approximate adders and multipliers [28, 36, 26, 30, 21, 27].

We focus on approximate multipliers; Horowitz [19] finds that multipliers are 7-10× more energy intensive than comparable adders. Mrazek et al. [31] and Sarwar et al. [37] discover an array of approximate multipliers using search methods such as Genetic Programming. We synthesize approximate accelerators using the open-source bank of approximate multipliers from Mrazek et al. [32].

Approximate deep learning: The resiliency of deep learning models to noise [4] motivates the design of approximate hardware for deep learning. Mrazek et al. [34] substitutes a single layer from an 8-layer ResNet with one utilizing an approximate multiplier. This work does not consider the effects of cross-layer approximation. ALWANN [33] searches for a mapping of layers to one of several fixed approximate units. However, ALWANN requires a model weight adjustment step to recover lost accuracy from approximation. Moreover, this fine-tuning step was not sufficient to avoid an accuracy degradation (e.g. a cited 0.6% degradation in accuracy for ResNet-50). Our method require no model fine-tuning step; therefore, it can be applied to models without modification. We also evaluate approximate designs on the large-scale ImageNet dataset; ALWANN evaluates on CIFAR-10 at low-resolution. Finally, we report real system-wide power numbers. We find in Figure 6 that it is necessary to evaluate whole system power rather than multiplier-only energy.

Approximation with compression: A large body of work has focused on compression techniques [2], including quantization [13], pruning [14, 9] and low-rank matrix decomposition [7]. We provide quantization techniques as a baseline for comparison. Compression methods such as pruning, distillation, and matrix factorization are orthogonal to our proposed framework on approximate circuits and our approach can be readily applied to a compressed neural network.

III. AutoApprox: A design framework for zero-loss approximate DNN accelerators

AutoApprox is an automated design framework for systolic-array based DNN ASIC accelerators. Given an architectural template design and a set of deep neural network workloads, AutoApprox generates both a hardware design and a heterogeneous mapping of neural network layers to the generated hardware. AutoApprox is a full-stack framework (see system diagram in Fig. 3) as it benchmarks candidate designs post-synthesis and evaluate designs using end-to-end workload metrics like top-1 accuracy.

Below, we cover key system components: (a) architectural template, (b) systolic array code generation, (c) circuit simulation for accuracy estimation and (d) chip performance estimation. We separately describe the design of circuit search and the layer mapping in Section IV.

A. TPUv3-based architectural template

We consider a systolic-array based accelerator with a design based on the TPUv3 [23, 22]. The TPUv3 contains several large systolic arrays for efficient matrix-matrix multiplication. The systolic array consists of a two dimensional array of processing elements (PEs), each containing one or more MAC units and buffers for input operands and output results. Global activation and parameter memory is shared across all systolic arrays. Overall, this design is extremely energy efficient with its usage of SRAM as each weight memory access is amortized across hundreds of operations.

As a consequence of thermal limits, power dissipation largely dominates the total cost of ownership (TCO) of modern inference chips. Consequently, in many cases die area may be traded off for designs which improve energy efficiency. Along these lines, one optimization which has become increasingly popular as chips entered the *dark silicon* regime [15] has been to provision special functional units which are only active for specific workloads to improve energy efficiency for these cases, but are otherwise inactive [44].

Our proposed architecture, shown in Figure 1, replaces a single exact MXU with a bank of several variants of approximate MXUs. At runtime, inputs are routed to one of these units based on a precomputed mapping. If we retain the exact MXU, this strategy can utilze the exact MXU for non-approximate workloads, thereby guaranteeing correctness while enabling power-savings for error-tolerant workloads. This approach does not require major modification to compiler stacks; it simply requires the addition of a `ROUTE` operation.

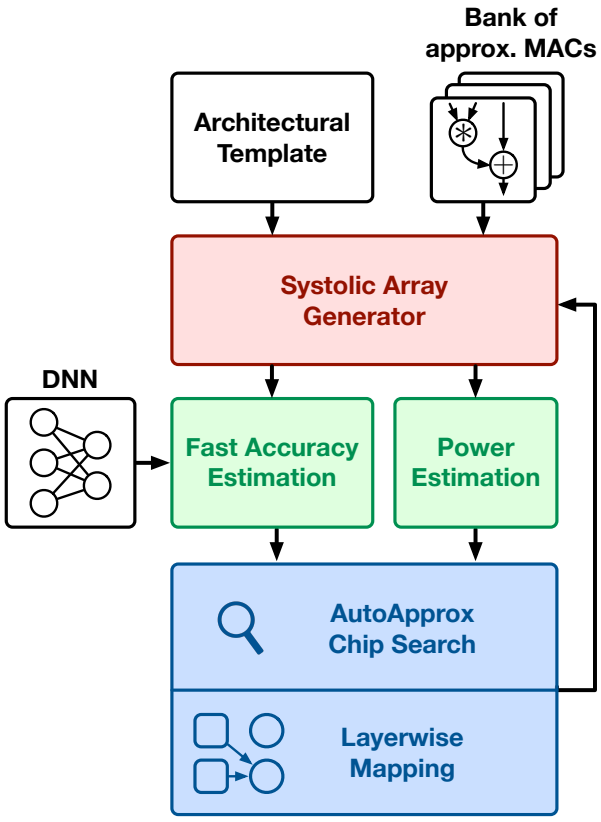

Fig. 3. Our system searches for low-power high-accuracy accelerators custom-tailored for a specific class of DNNs. At each iteration of search, a new approximate accelerator design is generated given an architectural template and a bank of approximate MACs. We estimate power using a commercial DC Topo tool. The end-to-end accuracy of the systolic array is benchmarked for a target DNN workload using our fast approximate circuit simulator.

MXUs that are not used in the computation for a specific layer are turned off to save power.

Arithmetic units in modern ML accelerators account for a large fraction of total system power while occupying the minority of the die [46, 22]. Therefore, our approach can dramatically reduce power consumption without chip area overhead. Our design is easy to deploy as it neither requires a new compiler stack nor control architecture. Our approach is also fully orthogonal to low-bit precision and can be combined to achieve further efficiency wins.

B. PE and interconnect generation

Given an architectural template and a list of candidate approximate multipliers, we generate code for the systolic array and accelerator. The list of approximate multipliers for the design comes from the AutoApprox ML-guided search procedure, described in Sec. IV. For each search iteration, a new set of candidate chips are generated. Code generation yields both Verilog and C++ implementations.

C. Fast accuracy estimation with Verilator

Both training and inference of large-scale deep neural networks entail trillions of arithmetic operations. With exact

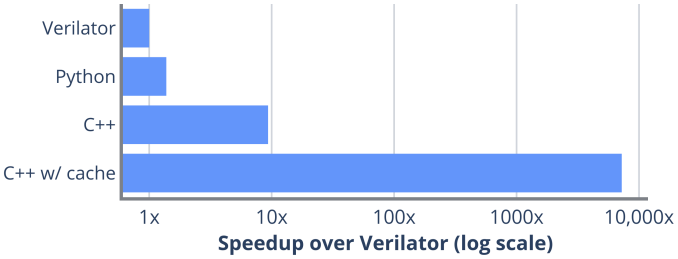

Fig. 4. By caching circuit evaluations, we accelerate the evaluation of approximate multipliers by 7200× over direct simulation with Verilator [39].

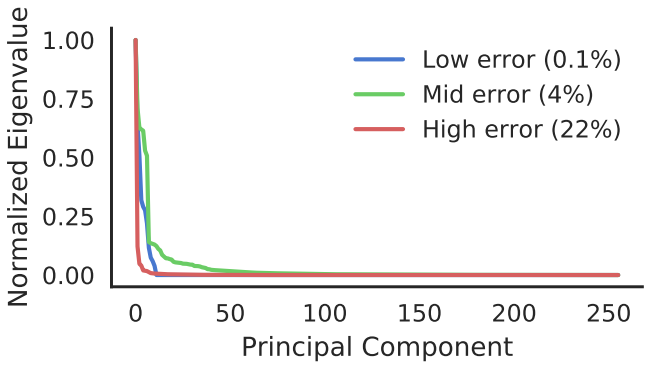

Fig. 5. Pre-computed outputs for approximate circuits are low-rank. We compress lookup tables to accelerate inference while reducing memory requirements from our simulator.

arithmetic and standard datatypes, this evaluation can be parallelized on high-throughput GPUs. However, we consider simulating inexact arithmetic with a custom MAC units. Direct evaluation of these inexact arithmetic operations is intractable since existing hardware does not support them. If we directly evaluate an approximate MAC with the Verilator circuit simulator [39], a single exact 8-bit multiplication takes 3.75 ± 0.95 microseconds on a high-end server. A single evaluation of ResNet-50 takes would 4.2 hours at 4 GFLOPs per 224x224 frame. For the entire ImageNet validation set, *evaluating a single approximate multiplier would take 23 years*.

Therefore, we cache calls to Verilator as in prior work [33]. However, Mrazek et al. [33] computes the full $2^{N \times N}$ lookup table for a N-bit multiplier. This does not scale to wide bitwidths; with 16-bit inputs, the look-up table would exceed 68 gigabytes. This precludes the use of GPU acceleration.

We find the lookup tables are low-rank and can be dramatically compressed without meaningful error using matrix decomposition. We include a visualization of the principle components for various approximate multipliers. We precompute the LUT in host CPU memory and then compute the low-rank eigen decomposition. We compute the approximation error matrix $\epsilon_{i,j} = \tilde{\text{m}}(i,j) - i \times j$ where $\tilde{\text{m}}$ is an approximate multiplier. In order to save memory when storing the N-bit error matrix $\epsilon \in \mathbb{R}^{2^N \times 2^N}$, we compute a truncated singular value decomposition with $k \ll 2^N$:

$$\epsilon \approx \sum_{i=1}^{k} \sigma_i u_i v_i^*$$

with total memory consumption of $O(nk)$, down from $O(n^2)$. With small $k \leq 50$, the total memory consmumption is under 20MB. During evaluation, we recompute the result of the approximate multiplication of $i \times j$.

We evaluate all results using the ImageNet 2012 dataset [5], a large-scale image classification dataset. We evaluate using a 10% sample of the full validation set in order to accelerate search by approximating the estimated accuracy of the end-to-end model on a target dataset. Overall, the sampled validation set contains 5000 images. Ranking models on this sampled validation set correlates with performance on the full dataset.

Overall, our optimizations result in a *7200× speedup* over direct circuit simulation in Verilator. This strategy also makes GPU evaluation feasible with future potential for automatic retraining. Accelerated evaluation of the approximate multipliers is critical to the feasibility of our search procedure. Compressing lookup tables with a low-rank decomposition should also enable our approach to scale to circuits at wider bitwidths in the future.

Our approach is complementary with state-of-the-art quantization methods. We perform dynamic range post-training quantization where weights are statically quantized to eight-bits prior to inference. During inference, activations are scaled to the `uint8` range of $[0, 255]$ and then quantized. We then perform dequantization with the procedure from Jacob et al. [20]:

$$\begin{aligned} q_3^{(i,k)} =& Z_3 + MNZ_1Z_2 - MZ_1 \sum_{j=1}^{N} q_2^{(j,k)} \\ & - MZ_2 \sum_{j=1}^{N} q_1^{(i,j)} + M \sum_{j=1}^{N} q_1^{(i,j)} q_2^{(j,k)} \end{aligned} \tag{1}$$

where q_1 represents the weight matrix, q_2 represents the activation matrix, and Z represents respective zero points. Higher power-savings could be accomplished with a more advanced quantization method utilizing quantization-aware training.

D. Performance estimation (power, area, delay)

Prior work primarily considers multilier-only performance metrics (e.g. power consumption of a single multiplier). However, we find a lower power multiplier does not necessarily result in a lower power systolic array in Figure 6. This is due to the impact of multiplier area on interconnect power; as area A increases, interconnect wire power must increase by $O(\sqrt{A})$. We perform all evaluation at a sub-10nm process as this is a leading technology node using Synopsys' physically-aware Design Compiler (Topographical) tool with a commercial PDK.

We assume a single clock domain in our architecture, whose frequency is dictated by the slowest MXU variant (typically the 'exact' MXU has the lowest intrinsic performance). As such during synthesis, the different MXU variants are constrained to

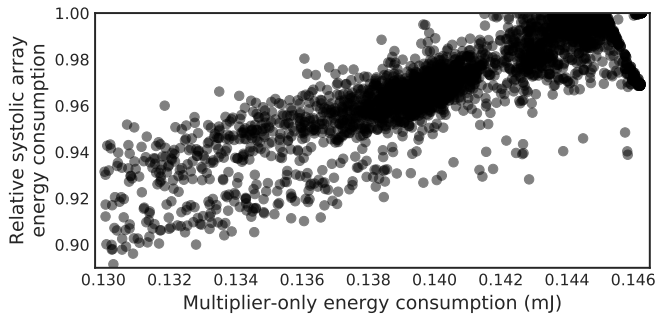

Fig. 6. The energy consumed by a single approximate multiplier and the energy consumed by the whole matrix multiply unit (MXU) are only weakly correlated. While more efficient multipliers can yield a more efficient chip overall, only considering the multipliers performance is insufficient when optimizing for whole chip power.

the slowest clock frequency – this enables some power and area savings as the synthesis tool is able to select smaller gate sizes for the approximate MXU variants which have higher intrinsic performance. In practice, DNN accelerators are TDP-limited rather than delay limited. Further gains are therefore possible by overclocking the chip using the thermal savings from the approximate systolic arrays; we evaluate the feasibility of this in Fig. 9.

The performance of a particular design also depends on how the convolution loop nest is mapped onto the array [35]. Mapping is important to ensure that generated accelerators match what would happen post-synthesis.

IV. Circuit search and layer mapping

In order to preserve high end-to-end task accuracy, we must carefully consider which portions of a workload are approximation tolerant. Dong et al. [8] find that the optimal approximation level changes dramatically between different layers of a neural network. We jointly consider the task of *selecting approximate units* for a chip design concurrently with the *mapping of layers* onto said chip's PEs. By co-designing the hardware and the mapping, we obtain accelerators that are custom-tailored to a particular class of neural networks.

However, each of these two subproblems are themselves challenging combinatorial optimization problems. Together, they represent a $O(K^N)$ search space with K candidate approximate multiplier designs and N neural network layers to map; we explore workloads with up to a 2^{268} search space. Therefore, random search will not perform well.

We leverage Bayesian optimization with GP bandits [29, 40] to efficiently discover high-accuracy yet energy-efficient configurations of cross-layer approximate circuits. This approach improves the sample efficiency of black-box optimization by modelling the unknown reward function $f : x \longrightarrow y$ with a Gaussian Process (GP).

A. Formalization of the approximate circuit mapping problem

Consider the following optimization problem to find the lowest power mapping of approximate circuits to deep network layers:

$$\min_{Z} \quad \sum_{i=1}^{N} q_i^\intercal Z_i \tag{2a}$$

$$\text{s.t.} \quad \text{ACC}(Z) f u \geq \tau \tag{2b}$$

$$\text{AREA}(Z) \leq \phi \tag{2c}$$

$$\sum_{j=1}^{K} Z_{ij} = 1 \qquad \forall i \in \{1, \ldots, N\} \tag{2d}$$

$$Z \in \{0,\, 1\}^{N \times K} \tag{2e}$$

The decision variable Z_i represents a one-hot vector to denote which of the K approximate circuits are mapped to layer i. The objective 2a models the total energy consumption to evaluate a single forward pass where $q_i \in \mathbb{R}_+^K$ represents a vector containing the energy to evaluate layer i for each of the K approximate multipliers. Constraints 2d and 2e ensure that Z_i is one-hot and is binary/integral. Constraint 2b defines a minimum accuracy target for the neural network. Finally, Constraint 2c constrains the area of the final chip to avoid degenerate solutions with many similar redundant multipliers.

The accuracy oracle ACC models the effect of cross-layer interactions from approximations. Given a particular assignment of approximate multipliers to layers, ACC calculates the expected accuracy of the model over a specific dataset. As errors introduced at one layer compound through subsequent layers, the accuracy oracle is exceptionally challenging to model. In this work, we evaluate the accuracy of a model over the validation set.

Unfortunately, this reduces the optimization problem to a black-box combinatorial optimization problem. To make matters worse, Bayesian optimization methods fail when applied to our optimization problem as they demonstrate slow convergence with a performance similar to random search. It is well known that Bayesian optimization struggles with high dimensional states [24], discrete structures [1] and constrained search spaces [11]. We therefore carefully reformulate our optimization problem to improve performance with off-the-shef Bayesian Optimization tools.

B. Calibration of single-layer approximation

To find high-accuracy approximate circuit designs, we constrain feasible solutions with a minimum validation accuracy threshold in Constraint 2b. However, the accuracy oracle ACC is not known. To reduce the complexity of the search space, we perform an offline study where we only use approximate multipliers for the target layer and evaluate all other layers with exact multipliers. This model provides an upper-bound on the expected accuracy from cross-layer approximation. We then prune mappings with exceptionally poor expected accuracy.

C. Continuous relaxation of state space

GP bandits are predominantly designed to optimize over discrete search spaces. Bayesian optimization frameworks typically support discrete search spaces by embedding them

in a real-valued box [12]. However, this embedding is sample-inefficient as it does not consider the relation between different categorical variables. For example, this solution has the challenge of instability due to quantization error from rounding continuous predicted variables to the nearest feasible points.

We utilize the estimate of per-layer accuracy degradation from single layer approximation (as described in the previous subsection) to compute an ordered set representing the relative ranking of each approximate multiplier. We define the ordering as the profiled end-to-end accuracy for approximating a single layer k with a particular multiplier. However, direct search with unrounded accuracy results in an unstable relaxed optimization problem. This results from two approximate multipliers which achieve similar accuracy, but have very different power consumption. We therefore relax the linear order of multipliers to a partially ordered set where ties within a fixed threshold of accuracy are considered incomparable. We then resolve a completed linear order by eliminating the least efficient multiplier in each pair of incomparable multipliers with similar accuracy.

This procedure derives a linear order of multipliers for each of the N layers in the neural network. To define distance in the new dimension after mapping, we apply min-max scaling to the resulting top-1 accuracy for each multiplier from single-layer approximation calibration. Given this new formulation of search space, we define the following cost optimization objective. For each of N layers, we define a step-wise cost function $Q_i : \mathbb{R} \longrightarrow \mathbb{R}$ to map a real-valued choice of an approximate multiplier (from 0 to 1) to the energy-consumption for the closest layer, rounding down.

The relaxed optimization problem is now:

$$\min_{z} \quad \sum_{i=1}^{N} Q_i(Z_i) \tag{3a}$$

$$\text{s.t.} \quad \text{ACC}(Z) \geq \tau \tag{3b}$$

$$\text{AREA}(Z) \leq \phi \tag{3c}$$

$$0 \leq Z \leq 1 \tag{3d}$$

$$Z \in \mathbb{R}^N \tag{3e}$$

D. Unconstrained optimization with barrier functions

While recent work has begun to explore multi-objective optimization using Bayesian optimization, these approaches are generally significantly less sample-efficient than single-objective optimizers. Ideally, we wish to explore the two-dimensional pareto frontier between accuracy and energy consumption. In practice, we find it useful to also limit the area of the final systolic array to avoid degenerate solutions where redundant approximate multipliers with similar accuracy are instantiated on a single chip.

We can utilize the barrier method [10] to remove constraints 3b and 3c. Barrier methods replace each constraint of form $x \leq b$ with a penalty in the objective function $\mathcal{B}(x, b) = -\log(b - x)$ or $\mathcal{B}(x, b) = e^{x-b}$. As x approaches the constraint b, the penalty trends to ∞. Utilizing a barrier method, we express our objective as:

$$\sum_{i=1}^{N} Q_i(Z_i) + \alpha_1 \mathcal{B}(\tau, \text{ACC}(Z)) + \alpha_2 \mathcal{B}(\text{AREA}(Z), \phi)$$

This objective now allows removal of constraints 3b and 3c. We leverage the exponential barrier function as it allows for soft constraint violations. For the accuracy term, we use a target accuracy $\tau = 0.68$ and weight $\alpha_1 = 8$. For the area term, we use a target area percentage (including exact multiplier) of $\phi = 400\%$ and a scale $\alpha_2 = 1.2$.

V. Experiments

We focus our evaluation on the ImageNet dataset [5], a large-scale dataset for image classification. ImageNet is a challenging dataset with 1M training images and 1000 classes, where each image is 224×224. Prior work has predominantly evaluated on the CIFAR-10 dataset. However, CIFAR-10 is a small dataset and not representative of modern computer vision workloads. Specifically, CIFAR-10 contains small 32×32 images, and only 10 classes that are well-separated. Because of this, approximation during neural network inference is expected to result in little change in overall class assignments.

We perform cross-layer search experiments using a 10% sample of the ImageNet validation set by evaluating 5 full-resolution images per class. This accelerates search without significantly impacting accuracy. On the validation set, the original model achieves 74.0% top-1 accuracy and 92.5% top-5 accuracy, while an 8-bit post-training quantized model achieves 72.1% top-1 accuracy and 90.7% top-5 accuracy.

We evaluate power savings and accuracy with the ResNet-50 architecture [17]. This model contains many interesting features that may affect approximation sensitivity, such as residual connections and batch normalization. As ResNet-50 is a deep network with many layers, it is a compelling target for studying cross-layer approximations and represents a practical application that is widely deployed today on production accelerators.

In searching for competitive architectures, we evaluate approximate multiplier variants from prior work [32]. We synthesized a total of 36 16×16 systolic array tiles for the different approximate multiplier variants in a commercial sub-10nm process using Synopsys' physical-aware Design Compiler (Topographical) tool, which provided performance, power, and area estimates. Evaluation of power, performance, and area of the systolic array – and not the base multipliers – is necessary to properly account for circuit and wiring overheads which temper some of the gains we would have observed if we only considered multiplier-level power, performance, and area. Finally, we assume a single clock domain shared by all approximate systolic arrays in our architecture, and as such scale power estimates for each approximate systolic array to the common clock frequency.

TABLE II

PARETO-OPTIMAL RESULTS FOR POWER, AREA AND ACCURACY ON IMAGENET VALIDATION SET FOR RESNET-50 WITH A SUB-10NM SYSTOLIC ARRAY HARDWARE ARCHITECTURE. POWER AND AREA ARE REPORTED RELATIVE TO THE ORIGINAL EXACT CIRCUIT (1.0× REPRESENTS THE EXACT QUANTIZED CHIP). AUTOAPPROX FINDS APPROXIMATE CIRCUITS THAT REDUCE ENERGY CONSUMPTION BY 3.2% AND AREA BY 5.2% WITH NO ACCURACY LOSS.

Hardware design	**Total chip energy** (relative to exact)	**Total chip area** (exact + approx)	**Top-1 accuracy**	**Top-5 accuracy**
Exact 8-bit MXU	1.0×	1.0×	72.1%	90.7%
Greedy layerwise search	0.976×	1.281×	71.2%	90.3%
Google Vizier [12]	0.969×	2.712×	65.82%	86.2%
AutoApprox-S (power optimized)	0.939×	1.844×	66.5%	87.42%
AutoApprox-L (balanced)	0.968×	0.948×	72.5%	90.7%
AutoApprox-XL (accuracy optimized)	1.024×	1.189×	73.1%	91.1%

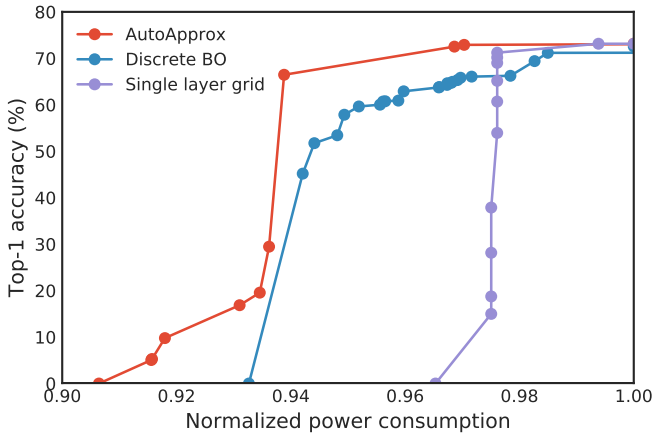

Fig. 7. AutoApprox finds more accurate approximate mappings at every power level relative to baseline approaches. The baseline Bayesian optimization framework fails to achieve zero-accuracy loss configurations while the grid search is unable to discover low-power designs.

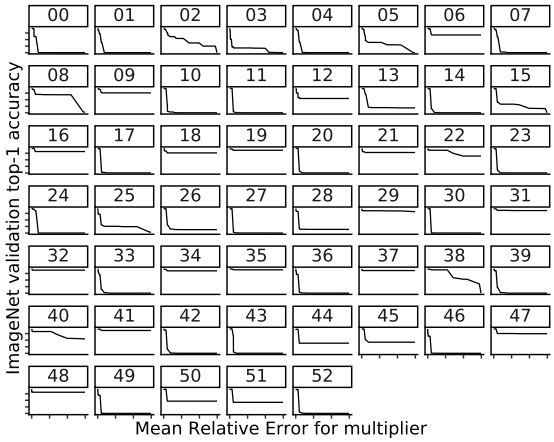

Fig. 8. Ablation of per-layer error tolerance for ResNet-50. Some layers are exceptionally robust to approximation (e.g. layer 31) but others are sensitive (e.g. layer 1). We utilize this sensitivity to compute a continuous relaxation of the discrete mapping search space.

VI. EVALUATION

A. How much power-savings can approximation achieve with minimal-to-no accuracy loss?

Since approximate arithmetic circuits can be integrated into hardware alongside quantization, we study how much additional *power is saved beyond quantization* and how much, if any, accuracy is lost from approximation. We benchmark results on the downsampled ImageNet validation set at full resolution with the ResNet-50 architecture. We compare against two key baselines: (1) a circuit using an exact 8-bit multiplier, (2) an exhaustive greedy baseline mapping a single layer to a single approximate multiplier, similar to the method proposed by Mrazek et al. [34] and (3) baseline search with the commercial black-box optimization toolkit Google Vizier [12]. We consider the Vizier baseline to perform similarly to Mrazek et al. [33] as both rely on combinatorial black-box optimization. For AutoApprox, we report three pareto-optimal designs: AutoApprox-S, AutoApprox-L and AutoApprox-XL. These configurations represent power optimized, a balanced and an accuracy optimized configuration.

We compare relative power consumption and area as well as validation accuracy in Table II. With no accuracy loss beyond the exact 8-bit circuit, AutoApprox discovers a circuit, labelled AutoApprox-L, with 3.2% lower energy consumption and 5.2% less circuit area. We also discover a lower-power approximate circuit with 6.1% less energy consumption, labelled AutoApprox-S; however, it degrades ImageNet validation top-1 and top-5 accuracy by 5.6% and 3.3% respectively.

Surprisingly, AutoApprox-XL discovers a configuration with 1.0% higher top-1 accuracy. We believe that approximations introduced by the custom-tailored circuit introduce regularization, similar to how pruning can improve generalization in the Lottery Ticket Hypothesis [9].

B. How does AutoApprox compare with other search methods?

We evaluate the energy-efficiency of our approximate circuit mapping procedures by examining the final Pareto-optimal trade-off curve between power and accuracy at the end of search. We run the baselines for a similar period of time (unless the method has a constrained search space). In addition to the greedy single layer heuristic, we evaluate a competitive black-box optimization framework [12].

Figure 7 compares the final power-accuracy Pareto curves for each method. AutoApprox discovers a higher accuracy

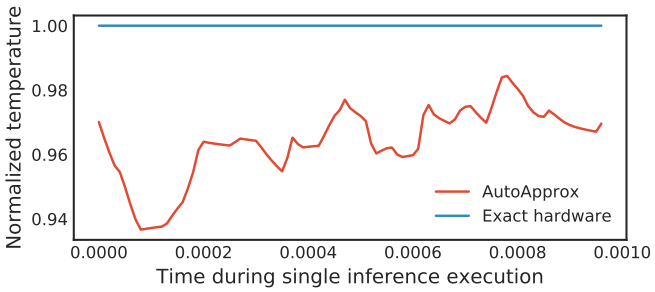

Fig. 9. The optimal design from AutoApprox generates less heat than a comparable exact circuit. Approximation enables TDP-limited chip designs to achieve higher performance via overclocking.

mapping than all baselines at every power consumption level. AutoApprox is able to maintain near-exact accuracy for many configurations. Moreover, AutoApprox is the only method to discover low-power chip designs. Greedy layerwise search is able to maintain high accuracies, but cannot discover low-power cross-layer approximate designs. Alternatively, the discrete Bayesian optimizer suffers a considerable accuracy loss due to the complexity of its search space. AutoApprox is able to find higher-quality solutions in significantly fewer iterations than baseline methods.

C. Ablation: error tolerance across ResNet-50 layers

Figure 8 displays the ImageNet validation set accuracy when substituting a single convolution (out of 52) with an approximate multiplier. On the horizontal axis, we rank each multipliers by the multiplier's expected mean relative error over a uniform distribution of input values. Some layers are exceptionally error tolerant (e.g. 31, 35 or 41). As more approximate multipliers are used, end-to-end accuracy does not drop significantly. However, certain layers (e.g. 1, 4 or 49) suffer dramatic accuracy drops as more approximate multipliers are used. This confirms that a layer-by-layer tuning approach must be used and that no one multiplier can be used homogenously across all layers of the neural network.

D. Ablation: Thermal behavior of the synthesized chip

In order to understand the characteristics of approximate circuits, in Figure 9, we take a single Pareto-optimal design and measure the normalized temperature of the chip using a vendor toolkit at a sub-10nm process node. While the discovered circuit had considerable power savings, it also operates at a significantly lower temperature when compared with exact hardware. This result opens the future opportunity to evaluate dynamically overclocking approximate circuits.

E. Discussion

Overall, AutoApprox is effective at discovering power-efficient approximate accelerators without accuracy loss. The results generalize to large-scale datasets and models. Moveover, this approach required no modification to the model's parameters and few changes to the runtime compiler stack. By routing layers approximate cores at runtime, we preserve the ability to run legacy workloads that may not be error tolerant.

Approximate computing has great potential to make a large dent in the efficiency of DNN accelerators. Conventional wisdom holds that approximation must result in a drop in accuracy. Surprisingly, we find the opposite; in some cases, custom-tailored approximations can increase accuracy. Optimizing inference accelerators by just a few percent (as found in our work) could have a large impact on cost and power consumption as model serving workloads far exceed training workloads in datacenters.

1) Other hardware architectures?: How well can we expect the results in this paper to generalize to novel hardware architectures? We leverage specific properties of the systolic array that simplify control via course-grained routing. Alternative architectures may require more complicated mechanisms to reconfigure the accuracy of the accelerator. However, as our methodology is fairly general, we expect the approach to generalize to new architectural templates.

2) Generalization to other DNNs?: How well will AutoApprox perform on novel architectures and datasets? We evaluated ResNet-50 as it is a standard benchmark for approximate DNN accelerators. However, nothing in our design is specific to convolutional neural networks. For example, the approximate systolic array design can be applied to improve the efficiency of the matrix multiplications in the Transformer [42] architecture. It is likely that the optimal set of approximate multipliers will be different between architectural families.

3) Future work: Our results could be further improved by retraining models to adapt to the chosen approximate circuits. In Figure 2, we see that approximate multiplier errors concentrate on specific inputs. Retraining would shift the distribution of DNN activations away from these error-prone values. Our results also considered a small library of open-source approximate circuits. Further gains could be realized by designing a novel approximate multiplier alongside mapping.

VII. Conclusion

We provide a framework, AutoApprox, that leverages approximate circuit design to generate energy-efficient inference circuits without any accuracy loss. Using a TPUv3 template design, we discover an efficient approximate accelerator that saves up to 3.2% of chip power consumption at zero-loss. By dynamically routing each layer of a neural network at runtime, we ensure only error-tolerant layers are routed to an approximate systolic array. Moreover, AutoApprox requires no major changes to the compiler stack thereby making deployment straightforward. We develop an scalable method that efficiently searches over the space of possible mappings of circuits to model layers with the goal of optimizing for energy performance and maintaining the end-to-end model accuracy. We demonstrate that we can substantially reduce the energy consumed on ImageNet dataset inference, without any degradation in accuracy. We hope this approach will enable improved efficiency for datacenter and edge inference.

References

[1] R. Baptista and M. Poloczek. Bayesian optimization of combinatorial structures. 2018.

[2] C. Chen, J. Choi, K. Gopalakrishnan, V. Srinivasan, and S. Venkataramani. Exploiting approximate computing for deep learning acceleration. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, 2018.

[3] P. Chi, S. Li, C. Xu, T. Zhang, J. Zhao, Y. Liu, Y. Wang, and Y. Xie. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News*, 2016.

[4] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan. Analysis and characterization of inherent application resilience for approximate computing. In *DAC*, 2013.

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009.

[6] L. Deng, G. Li, S. Han, L. Shi, and Y. Xie. Model compression and hardware acceleration for neural networks: A comprehensive survey. *Proceedings of the IEEE*, 2020.

[7] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus. Exploiting linear structure within convolutional networks for efficient evaluation. In *NeurIPS*, 2014.

[8] Z. Dong, Z. Yao, A. Gholami, M. W. Mahoney, and K. Keutzer. Hawq: Hessian aware quantization of neural networks with mixed-precision. In *CVPR*, pages 293–302, 2019.

[9] J. Frankle and M. Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks, 2019.

[10] R. Frisch. The multiplex method for linear programming. *The Indian Journal of Statistics (1933-1960)*, 1957.

[11] J. Gardner, M. Kusner, K. Weinberger, J. Cunningham, et al. Bayesian optimization with inequality constraints. In *ICML*, pages 937–945, 2014.

[12] D. Golovin, B. Solnik, S. Moitra, G. Kochanski, J. Karro, and D. Sculley. Google Vizier: A service for black-box optimization. In *SIGKDD*, 2017.

[13] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan. Deep learning with limited numerical precision. In *ICML*, 2015.

[14] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. In *NeurIPS*, 2015.

[15] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki. Toward dark silicon in servers. *IEEE Micro*, 2011.

[16] K. Hazelwood et al. Applied machine learning at facebook: A datacenter infrastructure perspective. In *HPCA*, 2018.

[17] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[19] M. Horowitz. Computing's energy problem (and what we can do about it).

[20] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam, and D. Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *CVPR*, pages 2704–2713, 2018.

[21] H. Jiang, J. Han, F. Qiao, and F. Lombardi. Approximate radix-8 booth multipliers for low-power and high-performance operation. *IEEE Transactions on Computers*, 2016.

[22] N. P. Jouppi, D. H. Yoon, G. Kurian, S. Li, N. Patil, J. Laudon, C. Young, and D. Patterson. A domain-specific supercomputer for training deep neural networks. *Commun. ACM*, 2020.

[23] N. P. Jouppi et al. In-datacenter performance analysis of a tensor processing unit. In *ISCA*, 2017.

[24] K. Kandasamy, J. Schneider, and B. Póczos. High dimensional bayesian optimisation and bandits via additive models. In *ICML*, 2015.

[25] Y. Kim, Y. Zhang, and P. Li. An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems. In *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013.

[26] C. Lin and I. Lin. High accuracy approximate multiplier with error correction. In *ICCD*, 2013.

[27] Z. Liu, A. Yazdanbakhsh, T. Park, H. Esmaeilzadeh, and N. S. Kim. Simul: An algorithm-driven approximate multiplier design for machine learning. *Micro*, 2018.

[28] H. R. Mahdiani, A. Ahmadi, S. M. Fakhraie, and C. Lucas. Bio-inspired imprecise computational blocks for efficient vlsi implementation of soft-computing applications. *Trans. Cir. Sys. Part I*, 2010.

[29] J. Mockus and L. Mockus. Bayesian approach to global optimization and application to multiobjective and constrained problems. *Journal of Optimization Theory and Applications*, 1991.

[30] A. Momeni, J. Han, P. Montuschi, and F. Lombardi. Design and analysis of approximate compressors for multiplication. *IEEE Transactions on Computers*, 2015.

[31] V. Mrazek, S. S. Sarwar, L. Sekanina, Z. Vasicek, and K. Roy. Design of power-efficient approximate multipliers for approximate artificial neural networks. In *ICCAD*, 2016.

[32] V. Mrazek, R. Hrbacek, Z. Vasicek, and L. Sekanina. Evoapproxsb: Library of approximate adders and multipliers for circuit design and benchmarking of approximation methods. In *DATE*, 2017.

[33] V. Mrazek, Z. Vasicek, L. Sekanina, M. A. Hanif, and M. Shafique. Alwann: Automatic layer-wise approximation of deep neural network accelerators without retraining. *ICCAD*, 2019.

[34] V. Mrazek, L. Sekanina, and Z. Vasicek. Using libraries of approximate circuits in design of hardware accelerators of deep neural networks. In *AICAS 2020*, pages 243–247, 2020.

[35] A. Parashar et al. Timeloop: A systematic approach to dnn accelerator evaluation. In *ISPASS*, 2019.

[36] K. M. Reddy, Y. B. Nithin Kumar, D. Sharma, and M. H. Vasantha. Low power, high speed error tolerant multiplier using approximate adders. In *VDAT*, 2015.

[37] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy. Energy-efficient neural computing with approximate multipliers. *J. Emerg. Technol. Comput. Syst.*, 2018.

[38] Y. S. Shao et al. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In *MICRO*, 2019.

[39] W. Snyder et al. Verilator, 2001. URL https://www.veripool.org/wiki/verilator.

[40] N. Srinivas, A. Krause, S. M. Kakade, and M. Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. *arXiv*, 2009.

[41] R. Vasudevan, D. Davydenko, and S. Skalicky. Model serving with amazon elastic inference, 2019.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need, 2017.

[43] S. Venkataramani, A. Ranjan, K. Roy, and A. Raghunathan. Axnn: Energy-efficient neuromorphic systems using approximate computing. In *ISLPED*, 2014.

[44] G. Venkatesh, J. Sampson, N. Goulding, S. Garcia, V. Bryksin, J. Lugo-Martinez, S. Swanson, and M. B. Taylor. Conservation cores: Reducing the energy of mature computations. ASPLOS, 2010.

[45] Q. Zhang, T. Wang, Y. Tian, F. Yuan, and Q. Xu. Approxann: An approximate computing framework for artificial neural network. In *DATE*, 2015.

[46] B. Zimmer et al. A 0.32–128 tops, scalable multi-chip-module-based deep neural network inference accelerator with ground-referenced signaling in 16 nm. *ISSCC*, 2020.