Learning to Design Accurate Deep Learning Accelerators with Inaccurate Multipliers

Paras Jain[1*], Safeen Huda[2], Martin Maas[2], Joseph E. Gonzalez[1], Ion Stoica[1] and Azalia Mirhoseini[2]
UC Berkeley[1], Google[2]
Corresponding author: parasj@berkeley.edu

***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 present 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 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 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, TPU**

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 [8]. Therefore, it is critical to improve the energy-efficiency of inference accelerators to reduce the global energy consumption demands of deep learning.

In systolic-array accelerators, Zimmer et al. [22] report that over 30% of PE energy is consumed by arithmetic units [17]. 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 [6]. However, low-bit quantization degrades accuracy [2]. The optimal precision for an architecture also varies widely between layers [3].

Emerging work proposes novel approximate circuits that are more power-efficient than quantized operators [1]. These approximate multipliers and adders do not simply reduce the bit-precision of exact arithmetic but rather *tailor approximations to a specific numerical distribution* observed in deployment. Approximate circuits thereby enable a better power-accuracy trade-off than quantization which uniformly approximates all inputs. Figure 2 shows the error for a single approximate multiplier. This multiplier concentrates error on select sparse values but consumes $\mathbf{3.61}\times$ less energy than an exact multiplier.

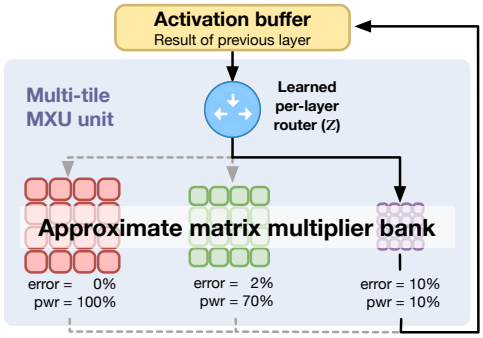

Figure 1: AutoApprox synthesizes approximate systolic-array accelerators without accuracy loss. Error-tolerant layers are routed to approximate cores for significant power savings.

Prior work in approximate computing for DNNs finds accuracy drops under errors [19, 14]. These approaches evaluate small-scale models and datasets. For example, Mrazek et al. [15] approximate just one layer of an 8-layer ResNet. Moreover, most prior work evaluates chip energy using the multiplier-only energy. However, we find that multiplier-only energy is only weakly correlated with total chip energy consumption as it excludes other factors (Figure 4).

To utilize approximate cores while preserving high accuracy, we take advantage of dark silicon by placing approximate systolic arrays adjacent to an optional exact array, as shown in Figure 1. At runtime, we dynamically route error-tolerant DNN layers to an 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 a modified TPUv3 template, AutoApprox generates a diverse set of efficient designs with a reconfigurable routing array to a bank of systolic arrays containing approximate multipliers [13]. By co-optimizing the mapping of approximate systolic arrays to layers, we avoid accuracy losses due to homogeneous approximation [3] by custom-tailoring the approximation for different fine-grained portions of the computation graph. We also perform evaluation on large-scale datasets and models, a first in the domain of functionally-approximate deep learning accelerators.

We evaluate AutoApprox with ResNet-50 models trained

[*]Work done while at Google

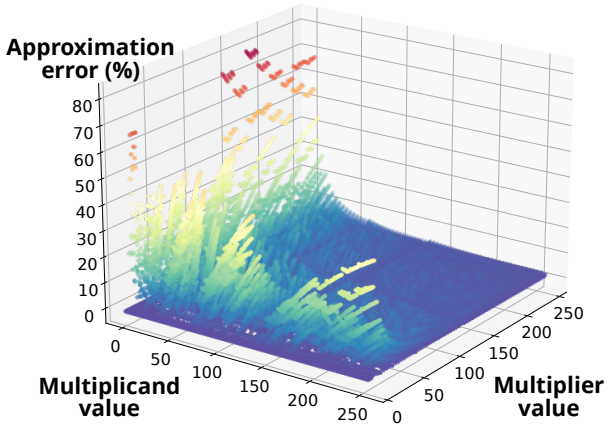

Figure 2: Approximate multipliers trade off exact computation with power efficiency. The visualized multiplier consumes 3.61× less energy than a quantized multipler at the cost of 4.2% relative error. By carefully matching approximate multipliers with each layer in a neural network, we are able to reduce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

on the ImageNet dataset as well as VGG-19 on the CIFAR-10 dataset. 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 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 by 7200×.
- In order to search the combinatorial space of accelerator designs, we develop a Bayesian optimizer to efficiently find low-power accurate designs.
- We demonstrate AutoApprox improves the power-efficiency of a TPU-based design without any added area while maintaining end-to-end accuracy.

I. Related Work

PARAS: Cite papers

Approximate computing: Analog computing offers large potential power savings; however, these methods are non-deterministic and therefore result in a large accuracy degradation while remaining hard to deploy. We instead focus on *functionally-approximate circuits* which replace power-intensive segments of a circuit with inaccurate but simpler components. There are many approximate arithmetic units; for example, prior work Kim et al. [12] removes overflow logic from a 16-bit adder for 2.3× multiplier-only savings. Specifically, we consider approximate multipliers; Horowitz [9] finds that multipliers are 7× more energy intensive than comparable adders. We synthesize approximate accelerators using the open-source bank of approximate multipliers from Mrazek et al. [13].

Table I: Overview of prior approximate computing methods for deep networks with comparison of key features

	Largest dataset	Model MACs	Retrain free?	Zero loss?
Venkataramani et al. [19]	CIFAR-10	<1M	✗	✗
Zhang et al. [21]	CALTECH	<1M	✗	✗
Sarwar et al. [16]	CIFAR-100	<1M	✗	✗
Mrazek et al. [15]	CIFAR-10	21M	✓	✗
Mrazek et al. [14]	CIFAR-10	120M	✓	✗
AutoApprox (ours)	ImageNet-1k	2B	✓	✓

Approximate DNN hardware: The of deep learning's resiliency to noise motivates the use of approximate hardware. Mrazek et al. [15] converts a single layer from an 8-layer ResNet to use an approximate multiplier. However, this work does not consider the effects of cross-layer approximation. ALWANN [14] searches for a mapping of layers to one of several fixed approximate units. However, ALWANN endures a 0.6% accuracy drop for ResNet-50 while our method results in no accuracy loss. 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. Figure 4 demonstrates the need to evaluate energy usage at the chip level rather multiplier.

II. AutoApprox: A Design Framework for Zero-Loss Approximate DNN Accelerators

AutoApprox is an automated design framework for systolic-array based DNN accelerators. AutoApprox is a full-stack framework (see system diagram in Figure 3) as it benchmarks candidate designs post-synthesis and evaluates designs using end-to-end workload metrics like top-1 accuracy. 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.

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 describe circuit search and the layer mapping in Section III.

A. TPUv3-based architectural template

We benchmark a production design based on the TPUv3 [10, 11] datacenter accelerator. The TPUv3 contains several large systolic arrays for efficient matrix-matrix multiplication. Each array is a 2D grid of processing elements (PEs) with one or more multiply-and-accumulate (MAC) units per PE. Global memory is shared across all systolic arrays.

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 utilize the exact MXU for non-approximate workloads, thereby guaranteeing correctness while enabling power-savings for error-tolerant workloads. This approach does not require modification to compiler stacks; it simply requires the

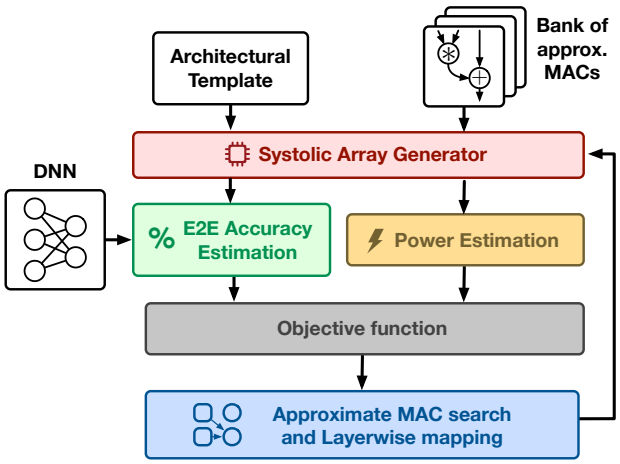

Figure 3: AutoApprox searches for low-power high-accuracy accelerators custom-tailored for a set of DNNs. Each search iteration generates a new approximate accelerator design given an architectural template and a bank of approximate MACs. Search minimizes energy subject to end-to-end accuracy constraints.

addition of a `ROUTE` instruction. Unused MXUs for a specific layer are turned off to save power.

Modern inference chips are limited by thermal dissipation. The *dark silicon* regime trades area for improved power efficiency where specialized functional units accelerate specific workloads but are otherwise inactive [7, 20]. In TPUv3, systolic arrays are extremely power-dense but only account for 20% of total die-area [11]. Therefore, our approach of adding auxiliary approximate systolic arrays promises to improve power efficiency with minimal die-area impact.

B. Systolic array generation

Given the TPU-v3 template and a list of candidate approximate multipliers, we generate any necessary systolic arrays and an accelerator. The list of approximate multipliers for the design comes from the AutoApprox ML-guided search procedure, described in Sec. III. Code generation yields both Verilog and C++ implementations.

C. Circuit simulation for end-to-end accuracy estimation

To precisely model the impact of approximation on end-to-end accuracy, we simulate hardware designs with sampled inputs from a target dataset. If we directly simulate an approximate MAC, a single exact multiplication takes 3.75 ± 0.95 microseconds on a high-end server while a single inference of ResNet-50 takes 4.2 hours and 23 years for all of ImageNet.

We leverage *caching* for memoization and *matrix decomposition* to make caching memory-efficient. Look-up tables (LUTs) have been used to simulate approximate circuits [15]; we use caching to accelerate simulation for 8-bit inputs by 7200x (Figure 5). However, storing full LUTs precludes GPU acceleration as two 16-bit inputs would require over 68GB. As shown in Figure 6, cache tables are low-rank. Thus, we can factorize the matrices without introducing significant error; when storing the N-bit error matrix $\epsilon \in \mathbb{R}^{2^N \times 2^N}$, we compute a truncated singular value decomposition with $k \ll 2^N$ with total memory consumption of $O(nk)$.

D. Performance estimation (power, area, delay)

Multiplier-only power is poorly correlated with total matrix-multiply unit (MXU) power in Figure 4. However, prior work solely considers multiplier-only power for simplicity. These designs are likely therefore suboptimal as they do not consider whole-chip power metrics and ignore power consumption due to interconnect and memory. By evaluating at a sub-10nm process using Synopsys' physically-aware Design Compiler (Topographical) tool, we ensure our designs correlate with real end-to-end power savings. We also assume a single clock domain in our architecture whose frequency is dictated by the slowest MXU variant on the chip. A single clock domain is conservative but simple to realize in actual hardware.

III. Approximate MAC 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. [3] find that the optimal approximation level changes dramatically between different layers of a neural network. To custom-tailor hardware to DNN approximation tolerance, we jointly consider the task of *selecting approximate units* for a chip design concurrently with the *mapping of layers* onto said chip's PE.

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.

We leverage Bayesian optimization with Gaussian Process (GP) bandits [18] 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 modeling the unknown reward function.

A. Formalization of the approximate circuit mapping problem

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

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

$$\text{s.t.} \quad \text{ACC}(Z) \geq \tau \tag{1b}$$

$$\text{AREA}(Z) \leq \phi \tag{1c}$$

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

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

The decision variable Z_i represents a one-hot vector to denote which of the K approximate circuits are mapped to a layer. The objective 1a 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 1b and 1c define

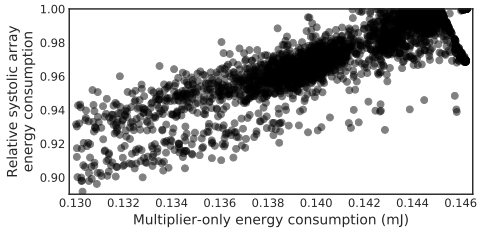

Figure 4: The energy consumed by a single approximate multiplier and the energy consumed by the whole matrix multiply unit (MXU) are only weakly correlated.

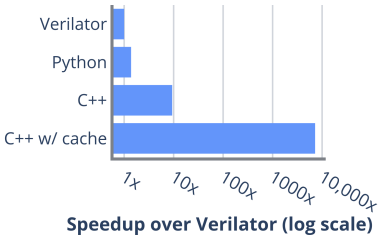

Figure 5: By caching circuit evaluations, we accelerate the evaluation of approximate multipliers by 7200× over direct simulation.

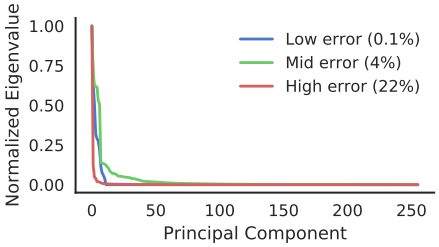

Figure 6: Pre-computed outputs for approximate circuits are low-rank. We compress lookup tables to accelerate inference while reducing memory requirements from our simulator.

the minimum accuracy targets and chip area limit respectively. Constraints 1d and 1e ensure that Z_i is one-hot and binary.

Bayesian optimization struggles with high dimensional states, discrete structures and constraints. In the following, we describe our approach to improve search performance via an unconstrained continuous reformulation.

B. Calibration of single-layer approximation

Implementing an efficient and accurate accuracy oracle ACC is extremely challenging due to cross-layer dependencies. To reduce the complexity of the search space, we perform an offline study where only one layer is approximate at a time. This single-layer calibration 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

Given the large number of states, Bayesian optimizers struggle to explore diverse solutions. We relax Z to a continuous variable which dramatically improves search sample efficiency. The calibration establishes an upper bound to the accuracy for a layer. The continuous relaxation searches per-layer for a target accuracy instead of a single categorical multiplier.

However, the objective function is no longer linear. Instead, 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 to the energy-consumption for the closest layer. The new objective is $\min \sum_{i=1}^{N} Q_i(Z_i)$.

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. We can utilize the barrier method [5] to remove constraints 1b and 1c. 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:

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

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 (including exact MXU) of $\phi = 400\%$ and a scale $\alpha_2 = 1.2$.

IV. Experiments

We focus our evaluation on the ImageNet dataset, a large-scale dataset for image classification. ImageNet is a challenging dataset with 1M training images and 1000 classes, where each image is resized to 224×224.

Prior work is predominantly evaluated on CIFAR-10. 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. ImageNet, however, is far more challenging as it contains many similar classes where approximation can blend the decision boundary between classes.

We evaluate power savings and accuracy with the ResNet-50 architecture. 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. To accelerate search, we perform cross-layer search experiments using a 10% sample of the ImageNet validation set.

In searching for competitive architectures, we evaluate approximate multiplier variants from prior work [13]. We synthesized a total of 36 16×16 systolic arrays 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.

As motivated by Figure 4, we evaluate power, performance, and area of the systolic array rather than the individual multipliers. Finally, we assume a single clock domain shared by all approximate systolic arrays in our architecture so we scale power estimates for each approximate systolic array to the common clock frequency.

Table II: Pareto-optimal results for power, area and accuracy on the ImageNet validation set for ResNet-50 with a sub-10nm TPUv3 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**
Baseline: Exact 8-bit MXU	1.000×	1.0000×	72.1%	90.7%
Greedy layerwise search	0.976×	1.281×	71.2%	90.3%
Bayesian optimization search using Vizier	0.969×	2.712×	65.8%	86.2%
AutoApprox-S (power optimized)	**0.939**×	1.844×	66.5%	87.4%
AutoApprox-L (balanced)	0.968×	**0.948**×	72.5%	90.7%
AutoApprox-XL (accuracy optimized)	1.024×	1.189×	**73.1%**	**91.1%**

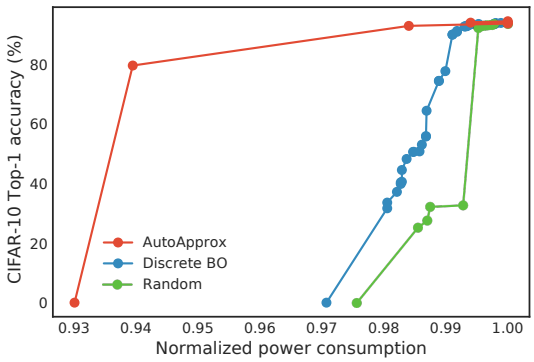

Figure 7: **VGG-19 on CIFAR-10** AutoApprox finds more accurate approximate mappings at every power level relative to baseline approaches. AutoApprox reduces end-to-end power consumption by 2% without accuracy loss.

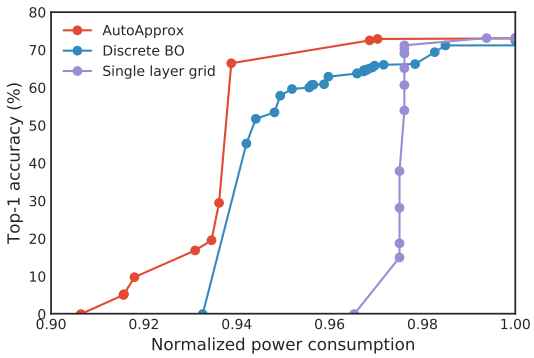

Figure 8: **ResNet-50 on ImageNet** AutoApprox scales to large models and datasets. The baseline Bayesian optimization framework fails to achieve zero-accuracy loss configurations while the grid search is unable to discover low-power designs.

V. 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 ImageNet validation set at full resolution with the ResNet-50 architecture. We compare against three 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. [15] and (3) baseline search with the black-box Bayesian optimization toolkit Vizier. We consider the Vizier baseline to perform similarly to Mrazek et al. [14] 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 a power optimized, a balanced and an accuracy optimized configuration.

We compare relative power consumption and area as well as validation accuracy in Table II. AutoApprox discovers a circuit, labeled AutoApprox-L, with 3.2% lower energy consumption and 5.2% less circuit area, while demonstrating higher accuracy than the quantized baseline. 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 hypothesize that approximations introduce regularization, similar to how pruning can improve generalization in the Lottery Ticket Hypothesis [4].

B. How does AutoApprox compare with other search methods?

We evaluate the energy efficiency of AutoApprox by examining the final Pareto-optimal trade-off between power and accuracy. We compare our approach to the greedy and Bayesian optimization search methods for a similar period of time.

Figure 7 compares the final power-accuracy Pareto curves for each method with a VGG-19 model trained over CIFAR-10. AutoApprox discovers a higher accuracy mapping than all baselines at every power consumption level. Notably, AutoApprox finds a high-accuracy design with 2% power savings end-to-end. We also compare against baseline search methods with the larger ResNet-50 model trained over ImageNet in Figure 8.

C. Ablation: Understanding AutoApprox mappings

Figure 9 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 inputs. Some layers are error tolerant (e.g. 31,

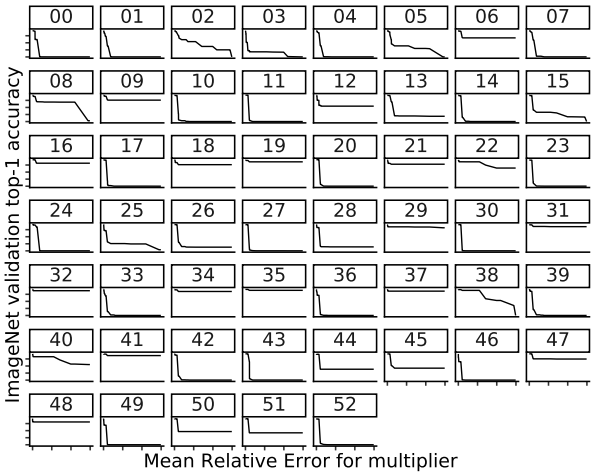

Figure 9: **Understanding AutoApprox mappings** Some layers are more robust to approximation (e.g. layer 31) than others (e.g. layer 1).

35 or 41). As increasingly approximate multipliers are used, end-to-end accuracy is preserved. However, certain layers (e.g. 1, 4 or 49) suffer dramatic accuracy drops. This observation further validates our layer-by-layer tuning approach.

VI. Conclusion

We propose AutoApprox, a framework 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 a scalable method that efficiently searches over the space of possible mappings of circuits to model layers with the goal of optimizing for energy while maintaining the end-to-end model accuracy. We demonstrate that we can substantially reduce the energy consumed on ImageNet inference at a sub-10nm process with no degradation in accuracy.

Acknowledgements

We thank Jeff Dean for inspiring us with the idea. We also thank Pulkit Bhuwalka, Christopher Clark, Anna Goldie, Jenny Huang, Ajay Jain, Azade Nazi, Nathan Pemberton, Suharsh Sivakumar, Ebrahim Songhori and Cliff Young for their support.

References

[1] C. Chen, J. Choi, K. Gopalakrishnan, V. Srinivasan, and S. Venkataramani. Exploiting approximate computing for deep learning acceleration. In *DATE*, 2018.

[2] 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.

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

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

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

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

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

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

[9] M. Horowitz. Computing's energy problem (and what we can do about it). In *ISSCC*, 2014.

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

[11] N. P. Jouppi et al. A domain-specific supercomputer for training deep neural networks. *CACM*, 2020.

[12] Y. Kim, Y. Zhang, and P. Li. An energy efficient approximate adder with carry skip for error resilient neuromorphic vlsi systems. In *ICCAD*, 2013.

[13] 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.

[14] 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.

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

[16] S. S. Sarwar, S. Venkataramani, A. Ankit, A. Raghunathan, and K. Roy. Energy-efficient neural computing with approximate multipliers. *JETC*, 2018.

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

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

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

[20] 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. In *ASPLOS*, ASPLOS, 2010.

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

[22] 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.