Evolving Energy Efficient Convolutional Neural Networks

Steven R. Young
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
youngsr@ornl.gov

J. Travis Johnston
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
johnstonjt@ornl.gov

Catherine D. Schuman
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
schumancd@ornl.gov

Pravallika Devineni
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
devinenip@ornl.gov

Bill Kay
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
kaybw@ornl.gov

Derek C. Rose
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
rosedc@ornl.gov

Maryam Parsa
Department of ECE, Purdue University
West Lafayette, Indiana, USA
mparsa@purdue.edu

Robert M. Patton
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
pattonrm@ornl.gov

Thomas E. Potok
Oak Ridge National Laboratory
Oak Ridge, Tennessee, USA
potokte@ornl.gov

***Abstract*—As deep neural networks have been deployed in more and more applications over the past half decade and are finding their way into an ever increasing number of operational systems, their energy consumption becomes a concern whether running in the datacenter or on edge devices. Hyperparameter optimization and automated network design for deep learning is a quickly growing field, but much of the focus has remained only on optimizing for the performance of the machine learning task. In this work, we demonstrate that the best performing networks created through this automated network design process have radically different computational characteristics (e.g. energy usage, model size, inference time), presenting the opportunity to utilize this optimization process to make deep learning networks more energy efficient and deployable to smaller devices. Optimizing for these computational characteristics is critical as the number of applications of deep learning continues to expand.**

***Index Terms*—neural networks, genetic algorithms, high-performance computing, energy efficiency**

I. Introduction

In the past decade, Deep Neural Networks (DNNs) have attracted great attention due to their promising results in various domains, including visual recognition [1], natural language processing [2], and artificial intelligence [3]. The general trend is towards creating deeper and more complicated architectures for achieving high accuracy, resulting in networks with large computation and storage requirements. For example, AlexNet [1], which achieved breakthrough results in the 2012 ImageNet Challenge contains 61 million parameters with 249MB of memory and 1.5B high precision operations to classify one image, while ResNet [4] proposed for the same task in 2015 has 50 convolutional layers with over 95MB memory for storage and over 3.8 billion floating point multiplications. Even models tailored for mobile deployment, such as SqueezeNet [5], MobileNet [6] and ShuffleNet [7] can be quite large.

To address the significant computational requirements of DNNs, several technology firms have created custom hardware like Google's Tensor Processing Unit [8], Intel Nervana [9], and IBM TrueNorth [10]. While most users of deep learning have similar energy efficiency or real-time processing needs, they will likely not have access to these advanced hardware systems, especially in the short term. Thus, rather than adapting the hardware to accommodate energy efficiency needs, we instead propose an approach for discovering deep learning *models* that are more energy efficient than their counterparts, through hyperparameter optimization.

In this work, we discuss an approach for that utilizes high-performance computing (HPC) to evolve the hyperparameters and topology of convolutional neural networks. We utilize an evolutionary approach that is capable of not only optimizing the hyperparameters of individual layers but also the type and arrangement of layers in the deep neural network [11]. This evolutionary approach allows us to explore a much larger search space of layer hyperparameter configurations

than is possible with neural architecture search (NAS) [12] while being more efficient than random search. We investigate the ability of this approach to produce energy efficient convolutional neural networks that can either be deployed at workstations or in data centers for more efficient, rapid data analysis. In particular, we analyze the performance of all of the networks created over the course of the hyperparameter optimization in terms of three metrics that are related to energy efficiency: inference time, model size, and energy consumption. We show that the different networks evolved can have radically different performance characteristics, and that as a byproduct of creating hundreds to thousands of different network topologies, we can choose our "best" network from among the evolved networks to suit our computational needs. The key conclusions of this work are:

1) Automated network design approaches that do not optimize for computational characteristics can have radically different energy consumption, model size, and inference time, and
2) There is an opportunity to utilize automated network design as an approach to creating more efficient networks in addition to current work in network pruning, etc.

II. Background and Related Work

There are a variety of approaches that are being developed to produce more energy efficient networks [13] including parameter pruning and sharing methods, quantization, low-rank approximation of layers, designing efficient hardware and other tailored optimization approaches.

Neural Network Optimization: Network pruning and sharing has been used both to reduce network complexity and to address the issue of over-fitting, and has been successfully applied to reduce resource requirements in convolutional networks [14], [15]. Le Cun *et al.* [14] introduced Optimal Brian Damage, which prunes weights with a theoretically justified saliency measure. [15] reduces network redundancy by identifying a subset of diverse neurons that do not require retraining, while [16] propose a data-free pruning method to remove redundant neurons. While pruning has been widely used, all pruning criteria require manual setup of sensitivity for layers, which demands fine-tuning of the parameters and could be cumbersome for some applications. DNN quantization [17] refers to the process of reducing the number of bits that represent a number. Model weights are often quantized so that parameters sharing the same value can be represented with fewer bits [18], [19]. In this way, the decision model represented by a neural network gets simpler and can be described using fewer number of bits, leading to better generalization abilities. Han *et al.* [20] used pruning to decrease connections by $9\times$, while quantization reduced the number of bits per connection from 32 to 5. As a result, the storage required from AlexNet dropped from 240MB to 6.9MB.

Low-rank factorization based techniques use matrix/tensor decomposition to estimate the informative parameters of the deep CNNs. Convolution operations correspond to bulk of the computation in deep CNNs. Rigamonti *et al.* [21] introduced the idea of learning separable 1D filters. [22], [23] used Canonical Polyadic (CP) decomposition for decomposing kernel tensors in convolutional layers and used iterative fine tuning to improve accuracy on the decomposed network. [24] proposed the use of tensor regression on fully-connected layers, showing an 80% space saving with only a 2% drop in ImageNet accuracy. While low-rank approaches are generalizable, they require layer by layer approximation with decomposition being a computationally expensive operation. While these approaches reduce the memory footprint, they do not necessarily translate into a reduction in energy consumption or deliver energy reduction that is proportional to the weight reduction. It is worth noting that, we do not use any of these approaches to improve performance. However, any of these approaches could be combined with our proposed approach to further improve performance.

Neural architecture search (NAS): NAS entails automatically designing effective neural network architectures with good generalization properties. NAS can be viewed as a hyperparameter optimization problem, since it requires setting a multitude of hyperparameters including an optimization method along with its associated set of hyperparameters, the dropout rate and other regularization hyperparameters [25]. The search spaces used for NAS are generally larger and control different aspects of the neural network architecture, the underlying problem is the same as hyperparameter optimization: find a configuration within the search space that performs well on the target task. Different search methods proposed for NAS include reinforcement learning [26]–[28], evolutionary algorithms [11], [29], sequential model based optimization [12], Bayesian optimization [30], [31], and Gradient-based optimization [32], [33].

[26] presented the first modern algorithm automating structure design, and showed that resulting architectures can indeed outperform human-designed state-of-the-art convolutional and recurrent networks. NAS problems are computationally intensive with results reporting hundreds or thousands of GPU-days for discovering state-of-the-art architectures [12], [26], [34]. In NAS via reinforcement learning [26], [27], a controller network is trained to sample promising architectures, while NAS with evolutionary algorithms evolves a population of networks for optimal DNN design [12], [35]. Recent work, ProxylessNAS, uses novel gradient-based frameworks by continuously relaxing of the search space to search the final models directly [36]. Our previous work, MENNDL [11] used evolutionary algorithms to produce hundreds of thousands of networks to find the most efficient network to optimize the given objective.

Custom hardware: Another approach for dealing with energy efficiency for deep learning systems is to use custom hardware. Google's Tensor Processing Unit (TPU) [8], custom accelerators now deployed in Google's data centers, can be used to achieve both significantly faster computation as well as significantly higher operations per watt. There are a variety of commercial products that are targeting faster, more energy efficient inferencing at the edge, including NVIDIA's

Jetson Nano [37], the Google's Edge TPU [38], and Intel's Movidius Neural Compute Stick [39]. However, these systems are typically not suitable for data center deployment, as they may be focused on real-time deployment (batch size of one). Neuromorphic systems, including Intel's Loihi [40], have also shown some promise for edge deployment with respect to energy efficiency and inference time [41]; however, neuromorphic computing systems are not readily available and sometimes require significant additional tuning in order to map deep learning networks to the neuromorphic architecture.

III. Methods

Multi-node Evolutionary Neural Networks for Deep Learning (MENNDL) is an algorithm and software package developed at Oak Ridge National Laboratory. MENNDL uses evolutionary algorithms and high-performance computing to evolve the topology and hyperparameters for convolutional neural networks. Previous iterations of MENNDL have been used to tailor convolutional neural network topologies for scientific datasets [11], [42]. MENNDL uses high-performance computing (HPC) to quickly evaluate many models. In particular, it uses a master-worker approach, in which the master runs an asynchronous genetic algorithm (GA) and the workers calculate the fitness of each candidate model. Calculating the fitness of the model requires training the candidate model, as well as calculating the accuracy (and other relevant objectives) on a validation set.

In all previous iterations of MENNDL, we simply took the best performing model in terms of accuracy on the validation set as the result of the hyperparameter optimization process. However, MENNDL creates hundreds to thousands of candidate models in just a few hours of computation time when using an HPC system. In this work, we analyze the resulting networks in terms of three additional evaluation metrics beyond accuracy, each of which is described below.

A. Evaluation Metrics

We evaluate the performance of each model not only in terms of accuracy on the testing set, but we also evaluate the performance in terms of three additional metrics: inference time, model size, and energy efficiency.

For inference time, we measure the amount of time required for the model to perform inferencing on the validation set of the dataset used. For this measurement, we run through the entire validation set of the dataset using a batch size of 100 and measure the total time required to obtain classifications for each image in the validation set. This timing includes the time required for data loading to the GPU. Because there is a fixed amount of time required for data movement, this provides a lower bound on the time required for inference.

The second performance metric we use in this work is model size. We use the number of parameters in the model as a proxy for the model size. As part of the evolution process, MENNDL can turn on and off layers. In this case, we used a maximum of up to 32 layers, where the first 16 layers can be one of convolution, inception, pooling, activation, or dropout, and the last 16 layers can be one of inner product, activation, or dropout. It is worth noting that a maximum of 32 layers is simply the chosen hyperparameter of MENNDL in this work. It can be extended to more layers or restricted to fewer layers as needed.

The third performance metric we use in this work is energy consumption. To obtain the energy consumption of the model, we use the command line tool nvidia-smi to periodically (once every 100ms) measure the GPU power usage. We then integrate that power measurement to obtain the energy usage. As in the case of inference time, we measure this value on the validation or testing set of the dataset used, but we divide by the number of images in the set to get an energy per inference.

B. Experimental Setup

We trained MENNDL on the Oak Ridge Leadership Computing Facility (OLCF) supercomputer Summit. Each of Summit's nodes has 2 IBM Power9 CPUs and 6 NVIDIA Volta GPUs. We used a subset of Summit's 4,608 nodes for training networks in this work. For each dataset, we performed 2 MENNDL runs on 24 nodes for 6 hours each.

For this work, we used the CIFAR-10, CIFAR-100, and CINIC-10 datasets. CIFAR-10 [43] is a multi-class dataset consisting of 60,000 32×32 color images in 10 mutually exclusive classes, with 6,000 images per class. CIFAR-10 is split into a training set of 50,000 images (5,000 of each class) and a testing set of 10,000 images (1,000 of each class). CIFAR-100 [43] is a multi-class dataset, again consisting of 60,000 32×32 color images, but with 1000 mutually exclusive classes. In this case, there are 600 images per class. CIFAR-100 is split into a training set of 50,000 images (500 per class) and a testing set of 10,000 images (100 per class). CINIC-10 [44] expands the CIFAR-10 dataset by including downsampled images from the ImageNet dataset [45]. Like CIFAR-10, CINIC-10 has 10 mutually exclusive classes, but it includes 270,000 32×32 color images. These are split into equally sized training, testing, and validation sets. For this work, we train using the training set, report the accuracy on the validation set, and do not use the testing set. It is important to note that we perform no dataset augmentation on any of these datasets, as we are simply performing a study to determine if there is an opportunity to optimize the computational characteristics of the networks, and we are not trying to create the best performing networks ever achieved for these datasets.

IV. Results

MENNDL produces hundreds to thousands of networks in each run at the scale used in these experiments. Each of those networks has different performance characteristics, though MENNDL's target is optimizing or evolving for accuracy.

Figure 1 gives a summary of the performance (in terms of accuracy and the three different computational metrics we are interested in evaluating) of different models that are generated over the course of MENNDL's evolution for CIFAR-10, CIFAR-100, and CINIC. In these plots, we highlight the

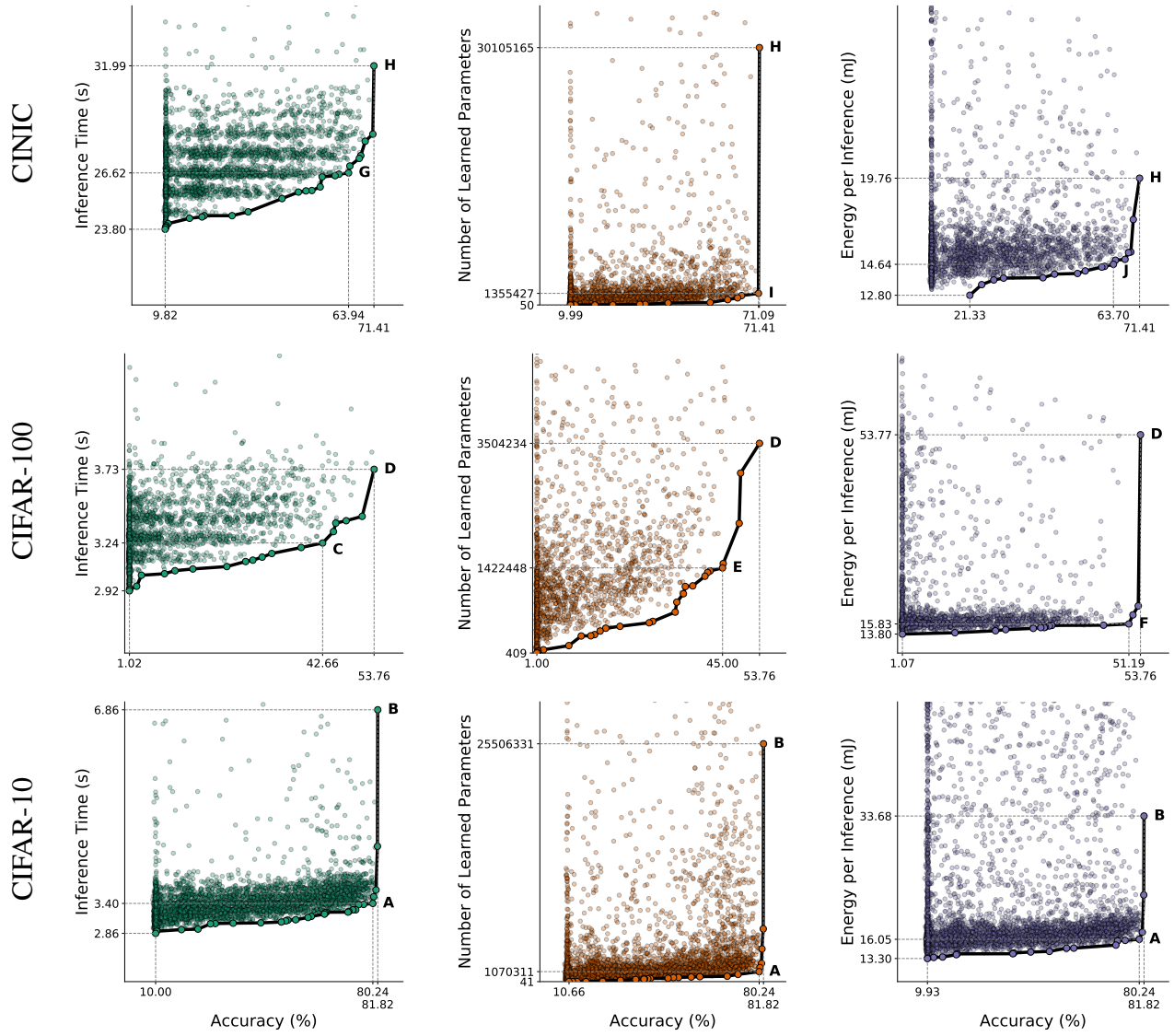

Fig. 1. Computational Characteristics (Inference Time, Number of Learned Parameters, Energy Consumption) vs. Accuracy for Each Dataset. Please note that these plots do not represent the entire range of values achieved for our computational metrics, but simply the range of interest for comparing the most accurate network and the networks with more desirable performance against these metrics.

TABLE I
PERFORMANCE SUMMARY OF MODELS

Dataset	CIFAR-10		CIFAR-100				CINIC			
Model	A	B	C	D	E	F	G	H	I	J
Inference Time (s)	3.39	6.85	3.23	3.72	3.31	3.41	26.62	31.99	28.56	26.77
Number of Learned Parameters	1070311	25506331	4422374	3504234	1422448	4079375	8109737	30105165	1355427	1005989
Energy per Inference (mJ)	16.05	33.68	15.72	53.77	36.41	15.83	25.90	19.755	42.03	14.64
Accuracy(%)	80.24	81.82	42.66	53.76	45	51.19	63.93	71.41	71.09	63.70

models that form the lower bound for each computational metric of interest in order to show the differences in performance and where trade-offs might be made. These highlighted models belong to the Pareto frontier of the problem, where each point represents a network in which it cannot be dominated by any other network in terms of optimizing both performance matrices. Network α dominates network β where dominance is defined by $\alpha \succ \beta$ or $\beta \prec \alpha$, iff $\forall i; f_i(\alpha) \leq f_i(\beta)$ (f_i is the i-th performance function) [46].

As can be seen in this figure, the performance for different models varies significantly, both in terms of accuracy as well as the other metrics we are interested in. In general, for each

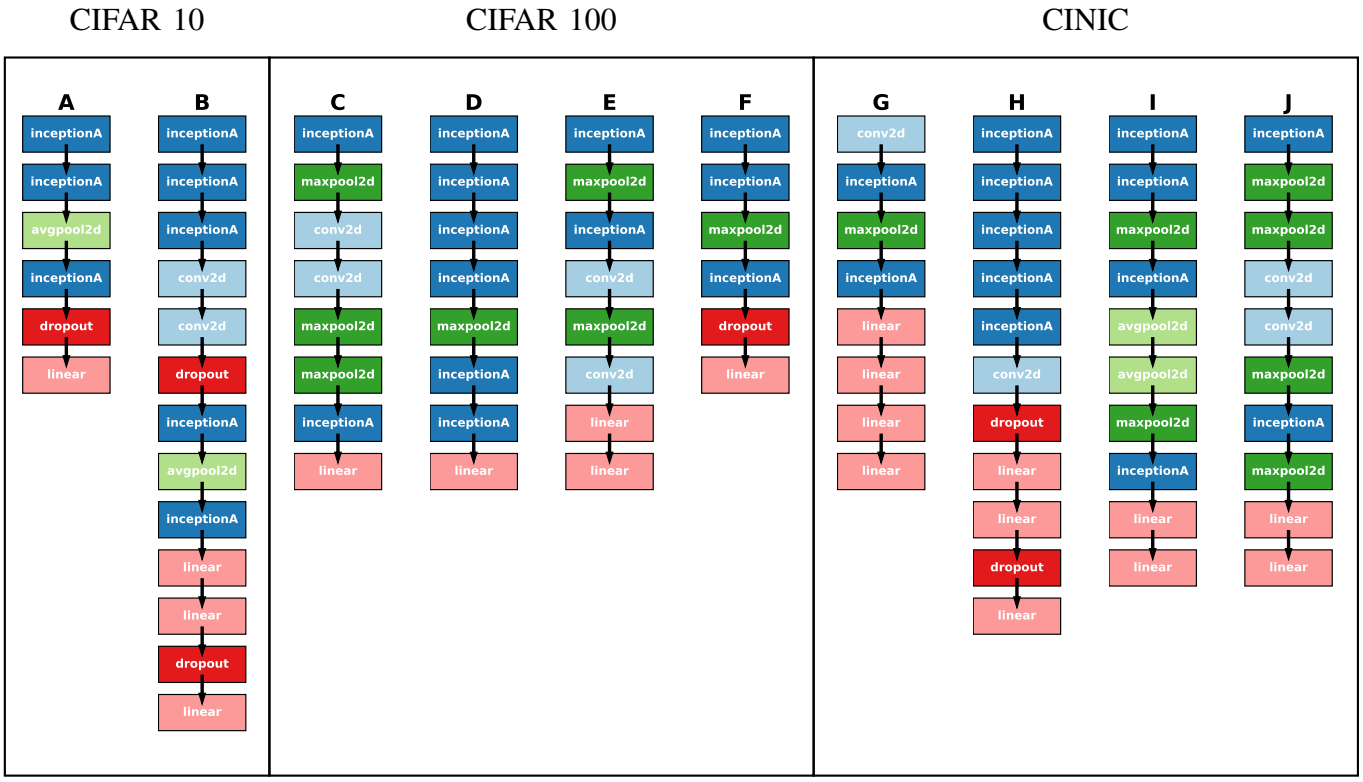

Fig. 2. Model topologies on the Pareto frontier referenced in Figure 1. Non-linear activations have been removed for clarity.

of the metrics, there is a steady increase in inference time, number of learned parameters, and energy to inference as the accuracy increases along the lower bound. This is consistent with what we would expect, which is that more complex models are required for higher accuracy networks.

In each of the plots, we highlight the accuracy and metric of interest for three models: the model with the lowest metric of interest (inference time, number of parameters, or energy to inference), the model with the highest accuracy, and the model that had the lowest weighted-Euclidean distance to the theoretical (but unattainable) best performance (0 inference time, energy or model size and 100 percent accuracy), which we call the "best trade-off" model. First, it is worth noting that all of the models with the lowest metric of interest have extremely low accuracies (often as bad as guessing on the task). Second, though there is some correlation between the three metrics of interest in terms of best performance for each of those three metrics, it is not always the case that the model with the lowest inference time is also the model with the fewest parameters and the model with the lowest energy consumption.

In Figure 1, the model with the highest accuracy in each plot and the best trade-off model are labeled with letters (A-J) to indicate which models are the same. Figure 2 shows the high-level model topology details of each of these models, though we omit the nonlinear activation layers. It is worth noting that the MENNDL process tunes the model topology and hyperparameters with no human intervention and relatively little restriction on what models "should" look like (models are sequences of layers where input/output shapes that computationally fit are chosen). Thus, hyperparameters may be set to non-traditional values (e.g., 20×3 kernel size) or non-traditional sequences of layers (e.g., model J in Figure 2 has two max pooling layers in a row). We have also not removed redundant computations; for example, MENNDL may produce networks that have multiple ReLU layers in a row that perform the same effective computation as a single ReLU layer. It may be possible to further improve the inference time of the models by eliminating redundant computations or condensing layers into one.

In the case of the CIFAR-10 dataset, the best model in terms of accuracy is model B and the beset trade-off model is model A, and in Figure 2 we can see that model B is significantly deeper than model A, but achieves very similar accuracy (model A has an accuracy of 80.24% and model B has an accuracy of 81.82%). In the case of CIFAR-10, the best trade-off model is the same for all three potential metrics of interest. This is not the case for the CIFAR-100 and the CINIC datasets. In Table I, we summarize the performance of each of the models A-J (labeled in Figure 1) for all of metrics of interest. In the case of CIFAR-10, model A is the best trade-off model for inference time, number of learned parameters, and energy to inference. In both CIFAR-100 and CINIC, however, there are three different best trade-off models (one for each of the metrics). Using this table, we can see that, for example, model I (the best trade-off model for number of parameters for CINIC) has comparable accuracy with the

most accurate model (H), but also has lower inference time. However, it has substantially higher energy than model H, so if energy is most important, it is likely not to be the best model to choose. Similarly, in the case of CIFAR-100, model F (the best trade-off model for energy) is comparable to the most accurate model (model D) for accuracy, but also has lower inference time. However, the number of parameters is substantially larger.

It is worth noting that using MENNDL takes significant computational effort to produce a well-performing model and that it may be counter-intuitive to use it to find a (set of) good models that can be used for more energy efficient computation. However, in terms of the life-cycle of the model, the majority of time will be spent performing inference on new images. Thus, it is likely worthwhile to spend the extra computational time during training to get to a better performing network during inference.

V. Conclusions and Future Work

The results clearly indicate that there is a huge opportunity to improve the computational performance of these networks if these characteristics are taken into account when performing automated network design. Even without steering the evolutionary process of MENNDL based on the computational characteristics of the networks, we see that there is often a $2\times$ improvement in energy usage, model size, and inference time between the networks with the highest accuracy and one that will perform almost as well. When it is expected for datacenter energy usage will account for 8% of energy usage worldwide in a decade [47], the ability to decrease the energy usage of a common workload will have a dramatic effect.

Figure 1 demonstrates the need to systematically search for the optimum neural architecture that maximizes all performance matrices such as accuracy, inference time, network size, and energy to inference. In future work, we will design a multi-objective optimization framework to perform the network search with minimum computational effort. We plan on using joint Bayesian and GA-based optimization platforms that are able to handle expensive black-box objective functions. This systematic approach is an essential step toward designing networks suitable for deploying on edge devices in resource-constrained environments. Additionally, we plan to combine this automated network design approach to creating efficient networks with compression, quantization, pruning, and/or low-rank approximations to push the bounds of efficient networks further. Finally, we also plan to use MENNDL to target the development of models specifically for edge deployment on specialized hardware.

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, under contract number DE-AC05-00OR22725. This research is sponsored in part by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[2] D. Amodei, S. Ananthanarayanan, R. Anubhai, J. Bai, E. Battenberg, C. Case, J. Casper, B. Catanzaro, J. Chen, M. Chrzanowski, A. Coates, G. Diamos, E. Elsen, J. H. Engel, L. Fan, C. Fougner, A. Y. Hannun, B. Jun, T. Han, P. LeGresley, X. Li, L. Lin, S. Narang, A. Y. Ng, S. Ozair, R. Prenger, S. Qian, J. Raiman, S. Satheesh, D. Seetapun, S. Sengupta, C. Wang, Y. Wang, Z. Wang, B. Xiao, Y. Xie, D. Yogatama, J. Zhan, and Z. Zhu, "Deep speech 2: End-to-end speech recognition in english and mandarin," in *Proceedings of the 33nd International Conference on Machine Learning, ICML 2016, New York City, NY, USA, June 19-24, 2016*, 2016, pp. 173–182. [Online]. Available: http://proceedings.mlr.press/v48/amodei16.html

[3] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, p. 484, 2016.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, 2016, pp. 770–778.

[5] 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," *arXiv preprint arXiv:1602.07360*, 2016.

[6] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[7] X. Zhang, X. Zhou, M. Lin, and J. Sun, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 6848–6856.

[8] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*. IEEE, 2017, pp. 1–12.

[9] Intel AI, "Intel Nervana Neural Network Processors (NNP) Redefine AI Silicon," https://www.intel.ai/intel-nervana-neural-network-processors-nnp-redefine-ai-silicon/, 2017.

[10] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam *et al.*, "TrueNorth: Design and tool flow of a 65 mw 1 million neuron programmable neurosynaptic chip," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 10, pp. 1537–1557, 2015.

[11] S. R. Young, D. C. Rose, T. Johnston, W. T. Heller, T. P. Karnowski, T. E. Potok, R. M. Patton, G. Perdue, and J. Miller, "Evolving Deep Networks using HPC," in *Proceedings of the Machine Learning on HPC Environments*. ACM, 2017, p. 7.

[12] C. Liu, B. Zoph, M. Neumann, J. Shlens, W. Hua, L.-J. Li, L. Fei-Fei, A. Yuille, J. Huang, and K. Murphy, "Progressive neural architecture search," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 19–34.

[13] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, "A survey of model compression and acceleration for deep neural networks," *arXiv preprint arXiv:1710.09282*, 2017.

[14] Y. LeCun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in neural information processing systems*, 1990, pp. 598–605.

[15] Z. Mariet and S. Sra, "Diversity networks," in *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. [Online]. Available: http://arxiv.org/abs/1511.05077

[16] S. Srinivas and R. V. Babu, "Data-free parameter pruning for deep neural networks," in *Proceedings of the British Machine Vision Conference 2015, BMVC 2015, Swansea, UK, September 7-10, 2015*, 2015, pp. 31.1–31.12.

[17] M. Wojnarski, "Nondeterministic discretization of weights improves accuracy of neural networks," in *European Conference on Machine Learning*. Springer, 2007, pp. 765–772.

[18] R. Xie, X. Jia, L. Wang, and K. Wu, "Energy efficiency enhancement for cnn-based deep mobile sensing," *IEEE Wireless Communications*, vol. 26, no. 3, pp. 161–167, 2019.

[19] A. Polino, R. Pascanu, and D. Alistarh, "Model compression via distillation and quantization," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018. [Online]. Available: https://openreview.net/forum?id=S1XolQbRW

[20] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[21] R. Rigamonti, A. Sironi, V. Lepetit, and P. Fua, "Learning separable filters," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2754–2761.

[22] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding-up convolutional neural networks using fine-tuned cp-decomposition," *arXiv preprint arXiv:1412.6553*, 2014.

[23] M. Astrid and S.-I. Lee, "Cp-decomposition with tensor power method for convolutional neural networks compression," in *2017 IEEE International Conference on Big Data and Smart Computing (BigComp)*. IEEE, 2017, pp. 115–118.

[24] J. Kossaifi, Z. Lipton, A. Khanna, T. Furlanello, and A. Anandkumar, "Tensor regression networks," *arXiv*, 2017.

[25] T. Elsken, J. H. Metzen, and F. Hutter, "Neural architecture search: A survey," *Journal of Machine Learning Research*, vol. 20, pp. 55:1–55:21, 2019. [Online]. Available: http://jmlr.org/papers/v20/18-598.html

[26] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[27] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le, "Learning transferable architectures for scalable image recognition," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 2018, pp. 8697–8710.

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

[29] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le, "Regularized evolution for image classifier architecture search," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 4780–4789.

[30] H. Jin, Q. Song, and X. Hu, "Auto-keras: An efficient neural architecture search system," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, 2019, pp. 1946–1956.

[31] K. Kandasamy, W. Neiswanger, J. Schneider, B. Poczos, and E. P. Xing, "Neural architecture search with bayesian optimisation and optimal transport," in *Advances in Neural Information Processing Systems*, 2018, pp. 2016–2025.

[32] S. Xie, H. Zheng, C. Liu, and L. Lin, "Snas: stochastic neural architecture search," *arXiv preprint arXiv:1812.09926*, 2018.

[33] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *International Conference on Learning Representations*, 2019. [Online]. Available: https://openreview.net/forum?id=S1eYHoC5FX

[34] E. Real, S. Moore, A. Selle, S. Saxena, Y. L. Suematsu, J. Tan, Q. V. Le, and A. Kurakin, "Large-scale evolution of image classifiers," in *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 2017, pp. 2902–2911.

[35] H. Liu, K. Simonyan, O. Vinyals, C. Fernando, and K. Kavukcuoglu, "Hierarchical representations for efficient architecture search," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. [Online]. Available: https://openreview.net/forum?id=BJQRKzbA-

[36] H. Cai, L. Zhu, and S. Han, "Proxylessnas: Direct neural architecture search on target task and hardware," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[37] NVIDIA, "Jetson nano," https://developer.nvidia.com/embedded/jetson-nano-developer-kit, cited September 2019.

[38] Google, "Edge TPU," https://cloud.google.com/edge-tpu/, cited September 2019.

[39] Intel, "Movidius neural compute stick," https://software.intel.com/en-us/neural-compute-stick, cited September 2019.

[40] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[41] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*. ACM, 2019, p. 1.

[42] S. R. Young, D. C. Rose, T. P. Karnowski, S.-H. Lim, and R. M. Patton, "Optimizing deep learning hyper-parameters through an evolutionary algorithm," in *Proceedings of the Workshop on Machine Learning in High-Performance Computing Environments*. ACM, 2015, p. 4.

[43] A. Krizhevsky, V. Nair, and G. Hinton, "CIFAR (Canadian Institute for Advanced Research)," 2009. [Online]. Available: http://www.cs.toronto.edu/ kriz/cifar.html

[44] L. N. Darlow, E. J. Crowley, A. Antoniou, and A. J. Storkey, "CINIC-10 is not ImageNet or CIFAR-10," *CoRR*, vol. abs/1810.03505, 2018. [Online]. Available: http://arxiv.org/abs/1810.03505

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[46] T. Okabe, Y. Jin, and B. Sendhoff, "A critical survey of performance indices for multi-objective optimisation," in *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, vol. 2. IEEE, 2003, pp. 878–885.

[47] A. Andrae and T. Edler, "On global electricity usage of communication technology: trends to 2030," *Challenges*, vol. 6, no. 1, pp. 117–157, 2015.