CS260 Final Project Report
Quantized ConvNets via Efficient Neural Architecture Search

Kai Wong (UID: 704451679)

kaileymon@g.ucla.edu

I. Introduction

This project is a qualitative study on the architecture design of quantized networks. Most of today's quantization or network compression techniques and algorithms are applied after a network is trained, yet model architectural decisions made for full-precision networks may not be ideal for their quantized counterparts. In this project, Efficient Neural Architecture Search (ENAS) was used to design convolutional cells (Pham et al., 2018) coupled with Mixed Precision Training (Micikevicius et al., 2018) in order to build convolutional architectures at half-precision (FP16) instead of the traditional single-precision (FP32) format. The models were trained on the MNIST dataset and the resulting architectures were compared against single-precision architectures designed under the same conditions using the same hyperparameters. The experimental results suggest that half-precision convolutional cells favor depth (via skip-connections in residual modules) over width (via wider inception modules) when the total effective number of parameters is kept constant.

II. Related Work

2.1 Neural Architecture Search

Most techniques explored in neural architecture search can be categorized into either reinforcement learning (RL) or evolutionary algorithm (EA) based methods. RL-based methods like NASNet (Zoph & Le, 2017) employ the use of a controller and operate over a discrete search space, typically designing networks at cell-level. EA-based methods like AmoebaNet (Real et al., 2018) do not employ the use of a controller but achieve results comparable to RL-based methods. More recent RL-based methods improve on the original NASNet algorithm by introducing strategies like sequential model-based optimization (SMBO) in the hierarchical approach taken by PNAS (Liu et al., 2018), or by incorporating transfer learning via parameter sharing to drastically speed up training, as described in ENAS (Pham et al., 2018).

Performing neural architecture search over continuous space has also been proposed such as in DARTS (Liu et al., 2018) where the search space is reformulated as a differentiable stochastic super-

net, with nodes representing intermediate data tensors like feature maps and edges representing operators like convolutions.

2.2 Neural Architecture Search with Quantization

There have been a few publications on using neural architecture search algorithms to design Convolutional Neural Networks (CNNs) with reduced precision or other quantization policies. Wu et al. (2018) proposed to perform neural architecture search over continuous space on fixed architectures in order to determine layer-wise precision assignments. Chen et al. (2018) proposed multi-objective neural architecture search for both CNN architecture and cell-wise quantization policy simultaneously. The approach was performed over a discrete search space using an evolutionary algorithm, much like the algorithm behind AmoebaNet (Real, Aggarwal, Huang & Le, 2018).

2.3 Mixed Precision Training

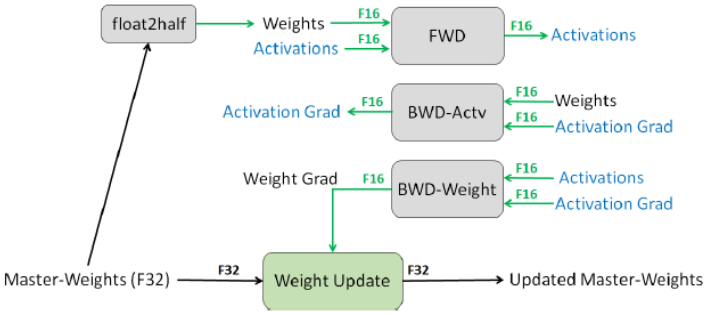


Figure 1: Mixed precision training iteration for a layer.
Mixed Precision Training, 2018. Retrieved: https://arxiv.org/abs/1710.03740

For quantization policies, this project only explores half-precision quantization via mixed precision training (Micikevicius et al., 2018). The methodology involves performing computes during training in half-precision while storing a master copy of variables in single-precision. Gradient calculations are also performed in half-precision, but loss-scaling is introduced to preserve gradient values with small magnitudes that may underflow. Gradients are then un-scaled by the same factor before performing weight updates on the FP32 variables in the master copy to preserve correctness.

III. Methods

This project combines ENAS with a half-precision quantization policy and the necessary training infrastructure to perform mixed precision training as described by Micikevicius et al. in order to compare learned CNN architectures. As noted by the authors of ENAS, parameter sharing between child models during the training stage might seem counter-intuitive since each child model might want to utilize their weights differently. However, this is encouraged by earlier work on transfer learning which established, with strong empirical evidence, that parameters learned for one task can be successfully used for another task by a different model (Razavian et al., 2014).

Apart from significantly less GPU hours required when using parameter sharing, another advantage specific to this project is that it restricts the decision space of the ENAS controller; because child models share the same set of weights, differences in the performance of the models would be (more or less) solely due to differences in architecture design choices made by the controller. This allows the experiments to better isolate the impact of quantization policies on the design of CNN architectures.

3.1 ENAS Overview

ENAS is capable of two methods of search: micro and macro search. Macro search is performed by defining a set of primitive operations like convolution or pooling and fixing the number of layers in the network, leaving the controller to choose operations layer-wise. Micro search on the other hand defines a network as being a sequence of blocks, where each block comprises N convolution cells and one reduction cell (reduction cells differ from convolutional cells only by their use of stride 2 in convolution and pooling operations). The controller then decides on operations cell-wise. This project currently only implements ENAS with micro search.

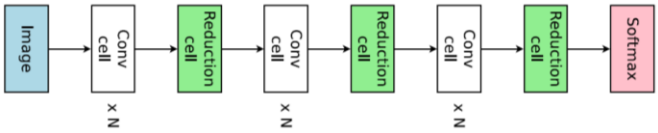


Figure 4. Connecting 3 blocks, each with N convolution cells and 1 reduction cell, to make the final network.

ENAS, 2018. Retrieved: https://arxiv.org/abs/1802.03268

ENAS proposes a stacked LSTM with 100 hidden units to function as the controller. In micro search, each cell is defined to have B nodes and is represented as a directed acyclic graph (DAG). Two nodes are designated as inputs, and for each of the remaining B − 2 nodes the controller decides on the following: 1) Two previous nodes that will be used as inputs for the current node, and 2) An operation that is applied to each of the two selected nodes. The outputs of the two operations are then added together and the value stored in the current node. The set of operations the controller can sample for each node consists of identity, separable convolution (5x5, 3x3), average

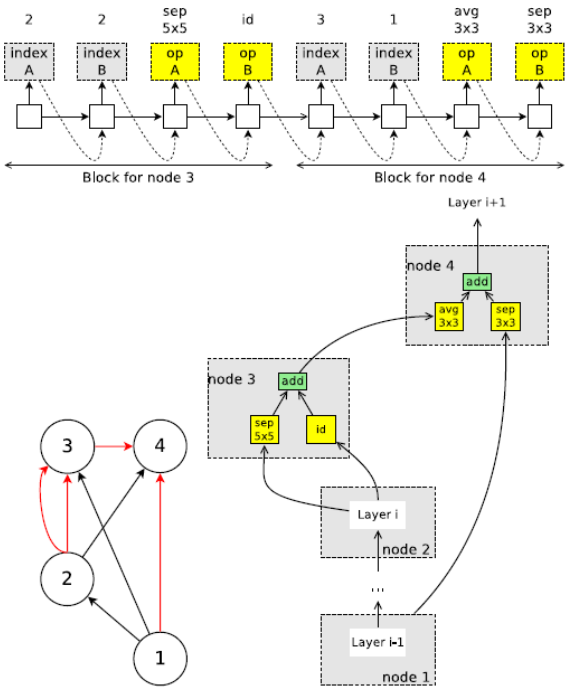


Figure 5. An example run of the controller for our search space over convolutional cells. *Top:* the controller's outputs. In our search space for convolutional cells, node 1 and node 2 are the cell's inputs, so the controller only has to design node 3 and node 4. *Bottom Left:* The corresponding DAG, where red edges represent the activated connections. *Bottom Right:* the convolutional cell according to the controller's sample.

ENAS, 2018. Retrieved: https://arxiv.org/abs/1802.03268

pooling (3x3) and max pooling (3x3) operations. More details on training specifics will be covered in the following subsection.

This project implements ENAS micro search with 5 and 7 nodes in each cell. Since the size of each cell is fixed, width versus depth trade-offs can be observed when child models are built in either single-precision or half-precision. Because the core focus of this project is to examine architectural differences at different precisions, a benchmark for validation accuracy is set and all child models that exceed the benchmark are sampled and compared. The experiments conducted in this project used a relatively low benchmark of 90.00% validation accuracy on the MNIST dataset for both single and half-precision model sets in order to obtain a larger sample of architectures for comparison.

3.2 Incorporating Mixed Precision Training

3.2.1 Loss Scaling

As suggested by Micikevicius et al. (2018), loss-scaling is essential when using mixed precision training in order to avoid gradient underflows. Loss-scaling works by scaling the loss value computed at the end of the forward pass. Subsequent gradient values computed during back-propagation are then scaled by the same factor by chain rule. Given a scaling factor $\boldsymbol{S}$, the new loss is given by $\boldsymbol{\mathcal{L}} \mapsto \boldsymbol{\mathcal{L}} \times \boldsymbol{S}$, and the computed gradient is $\nabla_{\omega}(\boldsymbol{\mathcal{L}} \times \boldsymbol{S}) = \boldsymbol{S} \times \nabla_{\omega}(\boldsymbol{\mathcal{L}})$. For small $\nabla_{\omega}(\boldsymbol{\mathcal{L}})$ below the minimum representable range of FP16, they would normally be lost as zeros. In many cases, this leads to model divergence (see Micikevicius et al., 2018).

Determining a suitable scaling factor was done as part of hyperparameter tuning for this project's experiments; selecting too small of a scaling factor would be ineffective but selecting too large a factor would result in overflows. Initially, this project intended to find $\boldsymbol{S}$ using an adaptive method where at each training iteration: $\boldsymbol{S}$ would be computed only when an underflow is detected using $\boldsymbol{S} = \left|\frac{\mathbf{65{,}504}}{\nabla_{\omega}(\boldsymbol{\mathcal{L}})}\right|$ since 65,504 is the maximum representable value in FP16. However, doing so incurred significant computational overheads and a fixed loss-scaling factor was opted for instead. This was determined empirically and a value of 5000 was found to be suitable.

3.2.2 ENAS Training

Like the ENAS training model described by Pham et al. (2018), learnable parameters in this project are categorized into two groups: Controller variables θ and the shared weights of child models ω. In lieu of mixed precision training, instead of having a FP32 master copy of ω for each child model generated at each training iteration, a single FP32 master copy ω would be shared among all child

models. This decision was again encouraged by previous work done on transfer learning (Razavian et al., 2014). Training can be further separated into two phases which are repeatedly alternated.

Phase 1: Training ω. Let $\mathbf{m}$ denote the set of all possible child models in a given cell, and $\boldsymbol{S}$ denotes the global loss-scaling factor. The policy of the LSTM controller $\pi(\boldsymbol{m};\theta)$ is fixed, and stochastic gradient descent (SGD) with Nesterov momentum is used to minimize the scaled expected cross-entropy loss on a training minibatch: $\min_{\omega} \mathbb{E}_{m\sim\pi(\boldsymbol{m};\theta)}[\boldsymbol{S}\times\mathcal{L}(\boldsymbol{m};\omega)]$. The scaled gradient is then computed using the Monte Carlo estimate given by:

$\nabla_{\omega}\mathbb{E}_{m\sim\pi(\boldsymbol{m};\theta)}[\boldsymbol{S}\times\mathcal{L}(\boldsymbol{m};\omega)] \approx \frac{\boldsymbol{S}}{M}\sum_{i=1}^{M}\nabla_{\omega}\mathcal{L}(\boldsymbol{m_i};\omega)$, where each $\boldsymbol{m_i}$ corresponds to a child model sampled from $\pi(\boldsymbol{m};\theta)$ during a training iteration. Following the authors, this project also uses $M = 1$, and ω is trained over a single training epoch each time. The learning rate is adjusted following a cosine schedule with $lr_{max} = 0.05$, $lr_{\min} = 0.001$, $T_0 = 10$ and $T_{mul} = 2$ (Loshchilov et al., 2017), along with a constant ℓ_2 weight-decay of 10^{-4}.

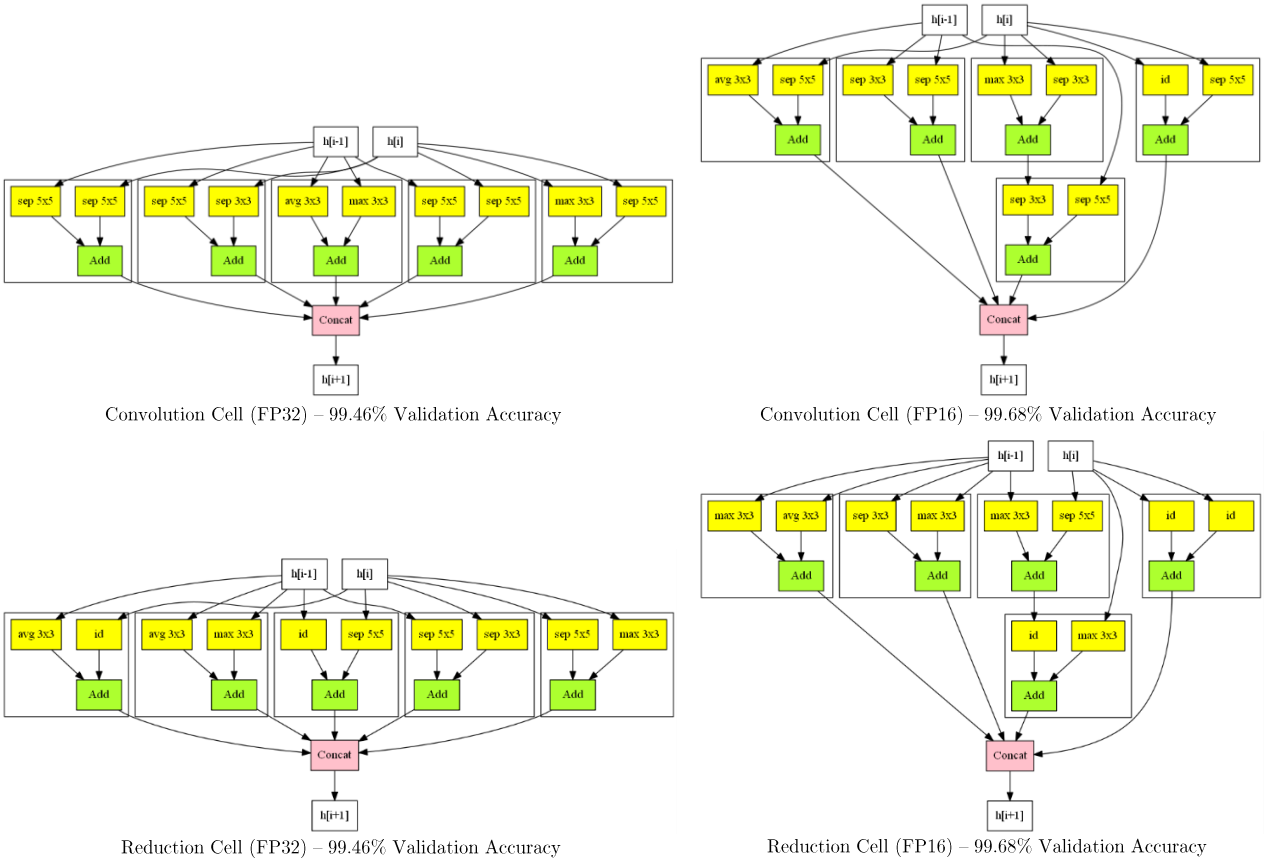
Phase 2: Training θ. Because this project is more focused on the derived CNN architectures, quantization is not applied to the LSTM controller. As such, training of the controller parameters θ follows that described by Pham et al. (2018): ω is fixed and the Adam optimizer with $lr = 0.00035$ is used to maximize the expected reward given by $\max_{\theta} \mathbb{E}_{m\sim\pi(\boldsymbol{m};\theta)}[\mathcal{R}(\boldsymbol{m};\omega)]$, where $\mathcal{R}(\boldsymbol{m};\omega)$ is computed as the highest accuracy among the sampled models on a validation minibatch. The gradient is computed using the policy gradient method described in REINFORCE (Williams, 1992) and updated at each time step t using the following: $\Delta\theta_t = \alpha\nabla_{\theta}\log\pi(\boldsymbol{m_t};\theta_t)\,\mathcal{R}_t$, $\alpha \sim Learning\ rate$.

IV. Experimental Results

Dataset: In all experiments conducted in this project, ENAS micro search was used to derive CNN architectures on the MNIST dataset. Some standard data pre-processing and augmentation was applied, including subtracting the channel (one channel in this case) mean, dividing the channel standard deviation, and random flipping of images.

Details: ENAS is applied to the micro search space with 5 nodes (blocks) in each cell for both single and half-precision settings. The experiments were performed using a single Nvidia GTX 1070 GPU for 100 epochs, where each epoch consisted of a single training cycle for the controller parameters θ and a single training cycle for the shared child model weights ω (section 3.2.2). Architecture data of child models generated by the controller which exceeded the 90.00% minibatch validation accuracy threshold were sampled for further processing.

4.1 Micro Search with 5-Node Cells



Convolution Cell (FP32) – 99.46% Validation Accuracy

Convolution Cell (FP16) – 99.68% Validation Accuracy

Reduction Cell (FP32) – 99.46% Validation Accuracy

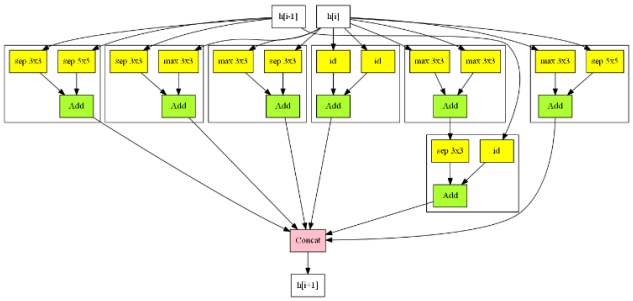
Reduction Cell (FP16) – 99.68% Validation Accuracy

Figure 6. Sample cells discovered for 5-node cells using ENAS on the micro search space for single and half-precision settings.

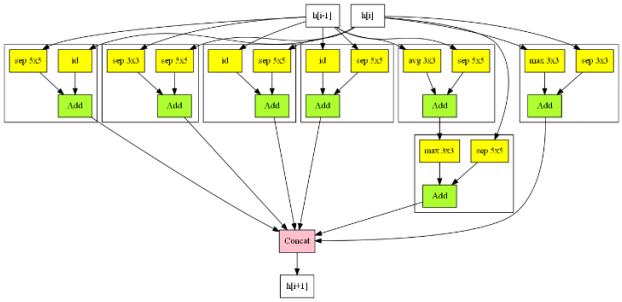
Figure 6 shows two sets of cells sampled from the experiments conducted in FP32 and FP16 respectively. In general, proposed cells uniformly converge to architectures that model inception modules (Szegedy et al., 2014). A possible reason for this may arise from using the validation set to compute the controller reward $\mathcal{R}(\boldsymbol{m}; \omega)$ as opposed to using the training set. This encourages ENAS to design architectures that generalize well rather than architectures that overfit the training set. Existing studies show that deeper networks are more prone to overfitting, hence ENAS converged to wider architectures instead. There was no noticeable difference in the operator compositions of either FP32 or FP16 sets.

In addition, skip-connections occur more frequently in the FP16 set, resulting in slightly less width versus depth in each cell for the same number of nodes. This result adds to the evidence that residual modules help alleviate the notorious problem of vanishing/exploding gradients (He et al., 2015). Vanishing gradients are still more likely to occur due to underflowing during mixed precision training of the FP16 child model set, even with the use of loss scaling. This could also serve as evidence for the need to improve on existing loss-scaling techniques used in conjunction with mixed precision training (Micikevicius et al., 2018). Table 1 summarizes CNN architectures designed by ENAS during the experiment.

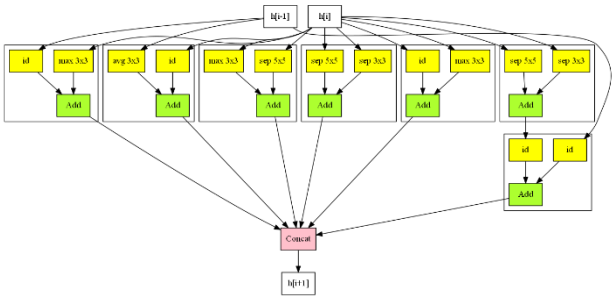
4.2 Micro Search with 7-Node Cells



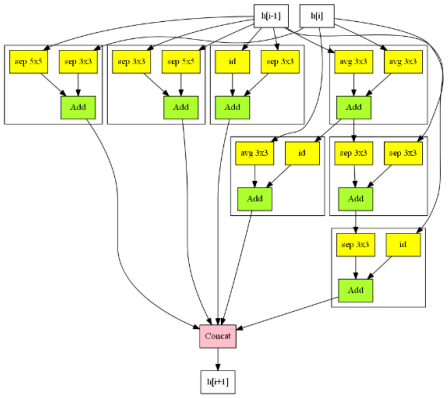
Convolution Cell (FP32) – 99.48% Validation Accuracy



Convolution Cell (FP16) – 99.22% Validation Accuracy



Reduction Cell (FP32) – 99.48% Validation Accuracy



Reduction Cell (FP16) – 99.22% Validation Accuracy

Figure 7. Sample cells discovered for 7-node cells using ENAS on the micro search space for single and half-precision settings.

When the experiment was repeated using 7 nodes per cell, there was an increase in the frequency of FP32 models having residual connections. Furthermore, especially for reduction cells, FP16 models exhibited denser residual connections in general.

	Total Samples	**Skip Frequency (%)**	**Average Skips/Cell**	**Conv to Pool Ratio**	**Average Width (nodes)**	**Average Depth (nodes)**
5-node (FP32)	731	0.21	0.34	2.13	4.81	1.19
5-node (FP16)	702	0.63	1.81	2.38	4.13	2.04
7-node (FP32)	653	0.54	1.63	2.21	6.08	2.11
7-node (FP16)	628	0.72	2.54	2.51	5.77	2.96

Table 1. Experiment statistics. *Total Samples* refers to the total number of child models proposed in the first 100 epochs that exceed the 90.00% validation accuracy threshold. *Skip Frequency* refers to the percentage of proposed cells containing at least one skip connection. *Average Skips/Cell* refers to the average number of skip connections in each cell. *Conv to Pool Ratio* refers to the average ratio of convolution to pooling operations in each cell. *Average Width* refers to the average width of cells in number of nodes. *Average Depth* refers to the average depth of cells in number of nodes.

V. Future Work

More extreme quantization policies using 8 bits, 4 bits, or even 1 bit might be worth exploring in this context and may uncover other architectural design aspects that are better suited for quantized CNNs. In the interest of time, the experiments in this project were not fully tested on CIFAR-10 or other more complex datasets but doing so in the future would be an important next step. Furthermore, the proposed child models with the highest minibatch validation accuracies could also be sampled and retrained from scratch to derive complete half-precision CNN architectures. The performance of these CNNs could then be compared against existing benchmarks for quantized image classifiers.

Recent work done on having different cell or layer-wise precision assignments (Wu et al., 2018, Chen et al., 2018) has provided strong empirical evidence for doing so. ENAS applied to the micro search space can be further improved to implement this idea by partitioning the shared weights ω of child models into independent cell-wise groups, which can then be used to determine independent cell-wise precision assignments. In addition, this project and the original ENAS restricts all cells to have the same number of nodes, which may not be ideal despite the good performance. The independent cell-wise partitioning could then be improved even further to allow for variable size selections.

VI. Conclusion

This project adds to the mounting evidence on the properties of residual connections and its ability to address vanishing gradients. At the same time, it also highlights the shortcomings of existing loss-scaling techniques used in mixed precision training which alleviate but do not fix the problem. Training using low-precision arithmetic can be challenging, and more work can be done to further refine the training methods. This is especially important in the context of learning on embedded platforms which have limited resources, where being able to train models directly with quantized weights would be significant (Li et al., 2017).

VII. References

A. Razavian, H. Azizpour, J. Sullivan, S. Carlsson. *CNN Features Off-The-Shelf: An Astounding Baseline for Recognition.* In *CVPR,* 2014.

B. Andrew, T. Lim, J. M. Richie and N. Weston. *SMASH: one-shot model architecture search through hypernetworks*. In *ICLR*, 2018.

B. Wu, Y. Wang, P. Zhang, Y. Tian, P. Vajda and K, Keutzer. *Mixed precision quantization of convnets via differentiable neural architecture* search. arXiv preprint arXiv:1812.00090, 2018.

B. Zoph and Q. V. Le. *Neural architecture search with reinforcement learning.* arXiv preprint arXiv:1611.01578v2, 2017.

C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich. *Going Deeper with Convolutions.* In *CVPR*, 2014.

E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. *Regularized Evolution for Image Classifier Architecture Search.* In *AAAI,* 2019.

H. Li, S. De, Z. Xu, C. Studer, H. Samet, T. Goldstein. *Training Quantized Nets: A Deeper Understanding.* In *NIPS*, 2017.

H. Liu, K. Simonyan and Y. Yang. *Darts: Differentiable architecture search.* arXiv preprint arXiv:1806.09055, 2018.

H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean. *Efficient neural architecture search via parameter sharing.* In *ICML*, 2018.

I. Loshchilov, F. Hutter. *SGDR: Stochastic Gradient Descent with Warm Restarts.* In *ICLR,* 2017.

K. He, X. Zhang, S. Ren, J. Sun. *Deep Residual Learning for Image Recognition.* In *CVPR,* 2015.

P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh and H. Wu. *Mixed Precision Training.* In *ICLR*, 2018.

R. J. Williams. *Simple Statistical Gradient-following Algorithms for Connectionist Reinforcement Learning.* In *Machine Learning,* 1992.

Y. Chen, G. Meng, Q. Zhang, X. Zhang, L. Song, S. Xiang and C. Pan. *Joint neural architecture search and quantization.* arXiv preprint arXiv:1811.09426, 2018.