Rethinking Differentiable Search for Mixed-Precision Neural Networks

Zhaowei Cai
UC San Diego
zwcai@ucsd.edu

Nuno Vasconcelos
UC San Diego
nuno@ucsd.edu

Abstract

Low-precision networks, with weights and activations quantized to low bit-width, are widely used to accelerate inference on edge devices. However, current solutions are uniform, using identical bit-width for all filters. This fails to account for the different sensitivities of different filters and is suboptimal. Mixed-precision networks address this problem, by tuning the bit-width to individual filter requirements. In this work, the problem of optimal mixed-precision network search (MPS) is considered. To circumvent its difficulties of discrete search space and combinatorial optimization, a new differentiable search architecture is proposed, with several novel contributions to advance the efficiency by leveraging the unique properties of the MPS problem. The resulting Efficient differentiable MIxed-Precision network Search (EdMIPS) method is effective at finding the optimal bit allocation for multiple popular networks, and can search a large model, e.g. Inception-V3, directly on ImageNet without proxy task in a reasonable amount of time. The learned mixed-precision networks significantly outperform their uniform counterparts.

1. Introduction

Deep neural networks have state-of-the-art performance on computer vision tasks such as visual recognition [16, 26, 29, 30, 12], object detection [24, 19, 5], segmentation [11, 6], etc. However, their large computation and memory costs make them difficult to deploy on devices such as mobile phones, drones, autonomous robots, etc. Low-precision networks, which severely reduce computation and storage by quantizing network weights and activations to low-bit representations, promise a solution to this problem.

In the low-precision literature, all network weights and activations are usually quantized to the same bit-width [14, 23, 35, 4, 37, 34]. The resulting *uniform* low-precision networks have been preferred mostly because they are well supported by existing hardware, e.g. CPUs, FPGAs, etc. However, uniform bit allocation does not account for the individual properties of different filters, e.g. their location

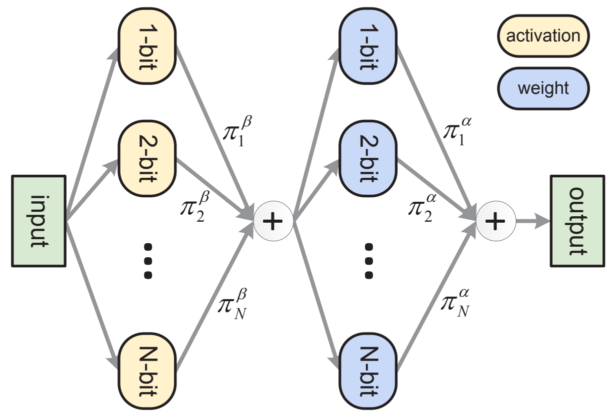


Figure 1. Differentiable architecture of the proposed mixed-precision network search module.

on the network, structure, parameter cardinality, etc. In result, it can lead to suboptimal performance for a given network size and complexity. *Mixed*-precision networks [1, 18, 17, 33, 32, 31, 9] address this limitation, enabling the optimization of bit-widths at the filter level. They are also becoming practically relevant, with the introduction of hardware that supports mixed-precision representations. For example, the NVIDIA AMP[1] can choose between different floating point representations during training.

Nevertheless, the problem of optimizing bit allocation for a mixed-precision network is very challenging. Since a network of L layers and N candidate bit-widths can have N^L different configurations, it is usually impossible to manually craft the optimal solution, and automatic bit allocation techniques are required. While this is well aligned with recent progresses in automatic neural architecture search (NAS) [38, 39, 3, 2, 20], there are several important differences between generic NAS and mixed-precision network search (MPS). First, NAS relies extensively on a proxy task to overcome the extremely high computational demands of searching for the optimal network architecture on a large dataset, such as ImageNet. It is common to search for a module block on a small dataset (such as CIFAR-10) and stack copies of this module as the final architecture. How-

[1] https://devblogs.nvidia.com/nvidia-automatic-mixed-precision-tensorflow

ever, this type of proxy task is very ineffective for MPS, due to 1) the layer importance difference, e.g. layers closer to input and output typically require higher bit-width; and 2) the (likely large) difference between optimal bit allocations for CIFAR-10 and ImageNet. Second, higher bit-widths usually lead (up to overfitting) to mixed-precision networks of higher accuracy. Hence, the sole minimization of classification loss usually has a trivial solution: to always select the candidate of highest bit-width. Third, while general NAS requires candidate operators that are heterogeneous in structure, e.g. convolution, skip connection, pooling, etc., MPS only involves homogeneous operators and very similar representations, e.g. convolutions of different bit-widths. These unique properties of MPS suggest the need for search techniques different than those of standard NAS.

In this work, we leverage the above properties to propose an efficient MPS framework based on several contributions. First, to enable search without proxies, the proposed framework is based on the differentiable search architecture of Figure 1, motivated by the popular DARTS [20] approach to generic NAS. Second, to avoid the trivial selection of the highest bit-width, learning is constrained by a complexity budget. The constrained optimization is reformulated as a Lagrangian, which is optimized to achieve the optimal trade-off between accuracy and complexity. Third, to circumvent the expensive second-order bi-level optimization of DARTS, a much simpler and effective optimization is proposed, where both architecture and network parameters are updated in a single forward-backward pass. Fourth, by exploiting the linearity of the convolution operator, the expensive parallel convolutions of Figure 1 are replaced by an efficient composite convolution, parameterized by the weighted sum of parallel weight tensors. This ensures the training complexity remains constant, independently of the size of search space, enabling the training of large networks.

Together, these contributions enable an efficient MPS procedure with no need for proxy tasks. This is denoted as *Efficient differentiable MIxed-Precision network Search* (EdMIPS) and can, for example, search the optimal mixed-precision Inception-V3 [30] of 93 filters, on ImageNet, in 8 GPU days. Extensive evaluations of EdMIPS on multiple popular networks of various sizes, accuracies, properties, etc., including AlexNet, ResNet, GoogLeNet and Inception-V3, show that it outperforms uniform low-precision solutions by a large margin. Beyond demonstrating the effectiveness of EdMIPS, this vast set of results also establishes solid baselines for the growing area of mixed-precision networks. To facilitate future research, all code is released at https://github.com/zhaoweicai/EdMIPS.

2. Related Work

Uniform low-precision: Low-precision networks have recently become popular to speed-up and reduce model size of deep networks [14, 23, 35, 4, 37, 34, 36]. [14, 23] pioneered the joint binarization of network weights and activations, using a continuous approximation to overcome the non-differentiability of quantization. This, however, entailed a significant accuracy loss. [35, 4] later achieved accuracies much closer to those of full-precision networks. The HWGQ-Net [4] approximated the ReLU by a half-wave Gaussian quantizer, and proposed a clipped ReLU function to avoid the gradient mismatch. LQ-Net [34] and PACT [7] tried to learn the optimal step size and clipping function online, respectively, achieving better performance. However, these are all uniform low-precision networks.

Bit allocation: Optimal bit allocation has a long history in neural networks [15, 1, 18]. [15, 1] proposed the exhaustive checking of filter sensitivities on a per layer basis, following simulation-based word-length optimization methods from signal processing [28]. [18] proposed an analytical solution to fixed-point quantization that seeks the optimal bit-width allocations across network layers by optimizing signal-to-quantization-noise-ratio (SQNR). [17] frames precision allocation as the sequential allocation of bits of precision to the weights of one layer, until a bit budget is exhausted. These techniques precede NAS [38] and are less powerful, less applicable to practical network design, or both.

Neural architecture search: NAS is a popular approach to automated search of neural network architectures [38, 39, 3, 2, 20]. However, the large and discrete nature of the search space make NAS very expensive. [38] proposed a reinforcement learning (RL) technique that requires 1,000 GPU days to search for an architecture on CIFAR-10. Subsequent works have tried to reduce these extraordinary levels of computation. [3, 2, 10] start by learning a supernet, containing all possible module choices, and find the best subnet within this supernet. Differentiable architecture search (DARTS) [20] relaxes the discrete search space into a continuous one, enabling the optimization by gradient descent.

Mixed Precision: Recently, [33, 32, 31] formulated MPS as an instance of NAS. Some of these techniques [33, 31] are based on RL and thus not very efficient. [32] relies on DARTS but requires a proxy task, due to the linear memory/computation on the cardinality of candidate pool, and samples dozens of architectures during search. Compared to these approaches, EdMIPS is simpler, more efficient, effective, and applicable to a much broader set of networks.

3. Low-Precision Neural Network

In this section, we briefly review some preliminary concepts on low-precision neural network.

3.1. Deep Neural Network

Deep networks implement many filtering operators,

$$y = f(a(x)) = \mathbf{W} * a(x), \tag{1}$$

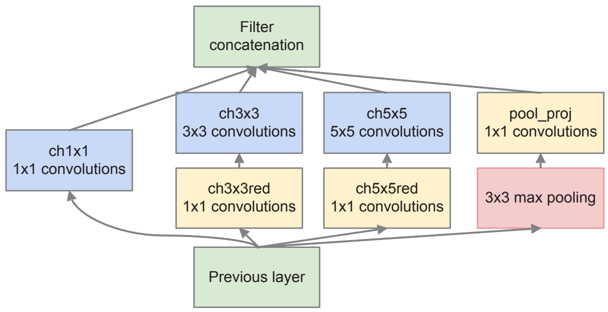


Figure 2. The Inception module in GoogLeNet.

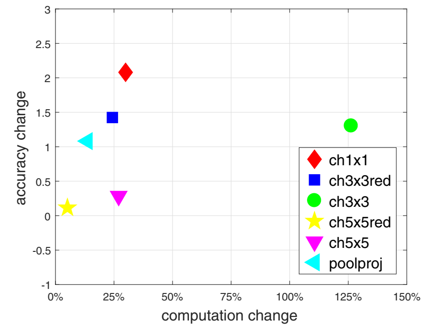


Figure 3. The filter sensitivity in the Inception module.

where filter f is parameterized by weight tensor $\mathbf{W}$ (which can be expressed in full or low-precision), x is the filter input, a a non-linear activation function (e.g. ReLU in full- or HWGQ [4] in low-precision networks), y the filter output and $*$ the convolution or matrix-vector multiplication operator. The L network filters $F = \{f_1, f_2, \cdots, f_L\}$ are learned by minimizing the classification risk $\mathcal{R}_E[F]$ defined by the cross-entropy loss function on a training set. The overall complexity of a neural network is mostly dictated by the complexity of (1). By quantizing both weights and activations to low bit-width, the expensive float-point operations of (1) can be replaced by efficient bit operations (e.g. XNOR and bit-count), substantially reducing model size and computation [23, 4].

3.2. HWGQ Network

A mixed-precision network can have arbitrary weight and activation bit-widths. Many quantization techniques have been proposed [14, 23, 35, 4, 37, 34]. In this work, we start from the HWGQ-Net [4], one of the state-of-the-art low-precision networks in the literature. Although the HWGQ-Net only uses binary weights, its activation quantization technique can be used to produce weights of higher precision. This starts by pre-computing the optimal quantizer

$$Q(x) = q_i, \quad if \quad x \in (t_i, t_{i+1}] \tag{2}$$

for a Gaussian distribution of zero mean and unit variance, using Lloyd's algorithm [21, 22]. Since the network weight distributions are always close to zero-mean Gaussians of different variance σ^2, the optimal quantization parameters can then be easily obtained by rescaling the pre-computed quantization parameters [27], i.e. using the quantizer

$$Q(x) = \sigma q_i, \quad if \quad x \in (\sigma t_i, \sigma t_{i+1}]. \tag{3}$$

To be hardware-friendly, all quantizers are uniform.

3.3. Filter Sensitivity

The allocation of the same bit-width to all network layers, known as uniform bit allocation, can be very suboptimal, since different filters have different bit-width sensitivities. Figure 3 illustrates this point for the popular Inception module of GoogLeNet [29] whose architecture is shown in Figure 2. This has four parallel branches and six learnable filters in total. To check their sensitivities, we started by training a uniform 2-bit GoogLeNet baseline on ImageNet. We then trained another model, changing a single filter of the Inception module to 4-bit, throughout the network. This experiment was repeated for all six filters. The changes in accuracy and computation of the whole network, with respect to the baseline, reflect the differences in bit-width sensitivity of the six filters. Figure 3 shows that "ch1x1" is the most sensitive filter, as the bit-width increase improves network accuracy by more than 2% with only a 25% computation increase. "ch3x3red" and "ch5x5" have computation similar to "ch1x1" but are less sensitive, especially "ch5x5". On the other hand, the computation of "ch3x3" skyrockets to 125% for a much smaller gain of 1.3% in accuracy, which is close to that of the inexpensive "ch3x3red" and "pool_proj". Finally, "ch5x5red" has minor changes in accuracy and computation. These observations suggest that mixed-precision networks could substantially outperform uniform networks.

4. Mixed-Precision Network

In a mixed-precision network, bit-width varies filter by filter. The problem is to search a candidate pool $\mathcal{B}$ of bit-widths for the optimal bit-width for each filter in the network. This can be implemented by reformulating (1) as

$$y = \sum_{i=1}^{n_f} o_i^\alpha f_i \left(\sum_{j=1}^{n_a} o_j^\beta a_j(x) \right), \tag{4}$$
$$\text{s.t. } \sum o_i^\alpha = 1, \sum o_j^\beta = 1, o^\alpha, o^\beta \in \{0, 1\},$$

where n_f and n_a are the cardinalities of bit-width pool $\mathcal{B}^\alpha$ for weights and $\mathcal{B}^\beta$ for activations. The goal is to find the optimal bit-width configuration $\{o_\alpha^*, o_\beta^*\}$ for the whole network. Since the search space is discrete and large, it is usually infeasible to hand-craft the optimal solution.

4.1. Complexity-Aware Learning

In general, networks of higher bit-width have higher accuracy. Hence, the simple minimization of classification loss has the trivial solution of always selecting the highest possible bit-width. To avoid this, we resort to complexity-aware learning, seeking the best trade-off between classification accuracy and complexity. This is a constrained optimization problem, where classification risk $\mathcal{R}_E[F]$ is minimized under a bound on complexity risk $R_C[F]$,

$$F^* = \arg\min_F R_E[F] \quad \text{s.t.} \quad R_C[F] < \gamma, \tag{5}$$

which can be solved by minimizing the Lagrangian

$$\mathcal{L}[F] = \mathcal{R}_E[F] + \eta\mathcal{R}_C[F], \tag{6}$$

where η is a Lagrange multiplier that only depends on γ. Under this constrained formulation, the optimal bit allocation is no longer trivial. The complexity is user-defined, and could address computation, memory, model size, energy, running speed, or other aspects. It has the form

$$\mathcal{R}_C[F] = \sum_{f\in F} c(f). \tag{7}$$

where $c(f)$ is the cost of filter f.

4.2. Model Complexity

A popular practice is to characterize complexity by the number of floating-point operations (FLOPs) of filter f,

$$c(f) = |f| w_x h_x / s^2, \tag{8}$$

where $|\cdot|$ denotes cardinality, w_x and h_x are the spatial width and height of the filter input x, and s the filter stride. In low-precision networks, where filter f and activation function a have low bit-width, this cost can be expressed in bit operations (BitOps),

$$c(f) = b_f b_a |f| w_x h_x / s^2, \tag{9}$$

where b_f and b_a are the bit-widths of weights and activations, respectively. Since only relative complexities matter for the search, the overall network complexity is normalized by the value of (8) for the first layer to search.

4.3. Relaxed Mixed-Precision Network

The binary nature of the search space of $\{o_\alpha, o_\beta\}$ makes the minimization of (6) a complex combinatorial problem. As suggested by [20], a much simpler optimization is possible by relaxing the binary search space into a continuous one, through the reformulation of (4) as

$$y = \sum_{i=1}^{n_f} \pi_i^\alpha f_i \left(\sum_{j=1}^{n_a} \pi_j^\beta a_j(x) \right), \tag{10}$$

$$\text{s.t.} \sum \pi_i^\alpha = 1, \sum \pi_j^\beta = 1, \pi^\alpha, \pi^\beta \in [0,1].$$

The constraints $\pi^\alpha, \pi^\beta \in [0,1]$ can then be eliminated by introducing a set of real configuration parameters $\{\alpha, \beta\}$ and defining

$$\pi_i^\alpha = \frac{\exp(\alpha_i)}{\sum_k \exp(\alpha_k)}, \quad \pi_j^\beta = \frac{\exp(\beta_j)}{\sum_k \exp(\beta_k)}. \tag{11}$$

This leads to the architecture of Figure 1. The complexity measure of (9) is finally defined as

$$c(f) = E[b_f] E[b_a] |f| w_x h_x / s^2, \tag{12}$$

where

$$E[b_f] = \sum_{i=1}^{n_f} \pi_i^\alpha b_{f_i}, \quad E[b_a] = \sum_{j=1}^{n_a} \pi_j^\beta b_{a_j} \tag{13}$$

are the bit-width expectations for weights and activations, respectively. This relaxation enables learning by gradient descent in the space of continuous parameters $\{\alpha, \beta\}$, which is much less expensive than combinatorial search over the configurations of $\{o_\alpha, o_\beta\}$.

4.4. Efficient Composite Convolution

While efficient, differentiable architecture search is not without limitations. A common difficulty for general NAS is the linear increase in computation and memory with the search space dimension [20, 2]. If there are ten candidate operators for a layer, they all need to be applied to the same input, in parallel. This makes the search impossible for large networks, e.g. the ResNet-50, Inception-V3, etc., and is usually addressed by resorting to a proxy task.

Unlike general NAS, where the candidate operators are heterogeneous [38, 20, 2], e.g. convolution, skip connection, pooling, etc., the candidate operators of MPS are homogeneous, namely replicas of the same filter with different bit-widths. This property can be exploited to avoid the expensive parallel operations. The weighted sum of parallel convolutions, as shown in Figure 1, given the weighted activation sum $\bar{a}(x) = \sum_{j=1}^{n_a} \pi_j^\beta a_j(x)$, can be rewritten as

$$\begin{aligned} y &= \sum_{i=1}^{n_f} \pi_i^\alpha f_i(\bar{a}(x)) = \sum_{i=1}^{n_f} \pi_i^\alpha (Q_i(\mathbf{W}_i) * \bar{a}(x)) \\ &= \left(\sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}_i) \right) * \bar{a}(x) = \bar{f}(\bar{a}(x)) \end{aligned} \tag{14}$$

where $\bar{f}$ is the composite filter parameterized by weight tensor

$$\overline{\mathbf{W}} = \sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}_i). \tag{15}$$

Hence, rather that n_f convolutions between the different $Q_i(\mathbf{W}_i)$ and the common activation $\bar{a}(x)$, the architecture of Figure 1 only requires one convolution with the composite filter of (15). This enables constant training time, independently of the number of bit-widths considered per filter, and the training of large networks become feasible.

In (15), each candidate operator has its own weight tensor $\mathbf{W}_i$. During training, the gradients arriving at a layer are distributed to the different branches. In result, filters of low probability π_i^α receive few gradient updates and could be under-trained. For example, a branch of $\pi^\alpha = 0.1$ will only receive 10% of the overall gradients. A more robust solution is to share weights across all filters by making $\mathbf{W}_i = \mathbf{W}$ and redefining the composite filter of (15) as

$$\overline{\mathbf{W}} = \sum_{i=1}^{n_f} \pi_i^\alpha Q_i(\mathbf{W}). \tag{16}$$

In this case, while the gradients are still distributed to each branch, they are all accumulated to update the universal weight tensor $\mathbf{W}$. This eliminates the potential for underfitting, and the size of search model can also remain constant independent of the search space. Note that this weight sharing requires a universal $\mathbf{W}$ representative enough for multiple quantizers of different bit-widths. While this has not been shown in the low-precision network literature [35, 4, 37, 34], it is less of a problem for MPS since 1) the parallel branches learn similar and potentially redundant representations at different bit-widths; 2) what matters for MPS is to optimize the bit allocation, not the weight tensor.

4.5. Search Space

The design of the search space is critical for NAS. Unlike NAS, which has an open-set search space, the search space of MPS is well-defined and limited to a relatively small number of possibilities per filter and activation, e.g. bit-widths in $\{1, \ldots, 32\}$. Previous works [32, 10] have coupled weight and activation bit-widths, e.g. defining pairs $(1, 4)$, $(2, 4)$, etc. This results in $|\mathcal{B}^\alpha| \times |\mathcal{B}^\beta|$ parallel branches per filter. To reduce complexity, [32, 10] manually pruned $|\mathcal{B}^\alpha| \times |\mathcal{B}^\beta|$ to a small subset, e.g. six pairs, which is suboptimal. Instead, we decouple weight and activation bit-widths, using $|\mathcal{B}^\alpha|$ and $|\mathcal{B}^\beta|$ parallel branches for weight and activations, respectively, as shown in Figure 1. This decoupling maintains the search space at full size $|\mathcal{B}^\alpha| \times |\mathcal{B}^\beta|$, but significantly reduces computation and memory. Since 1) many works [34, 7] have shown that a bit-width of 4 suffices for very good performance; 2) very low-precision networks, e.g. using 2 bits, are the most challenging to develop; and 3) 1-bit activations are usually insufficient for good performance [14, 23, 35, 4], we use a search space with

$$\mathcal{B}^\alpha = \{1, 2, 3, 4\}, \quad \mathcal{B}^\beta = \{2, 3, 4\}. \tag{17}$$

4.6. Learning

This leads to the learning procedure for MPS. EdMIPS optimizes the Lagrangian of (6) with respect to the architecture parameters $\{\alpha, \beta\}$ of (11) and the weight tensors $\mathbf{W}$ of (16) over the search space of (17). Both $\mathbf{W}$ and $\{\alpha, \beta\}$ are learned by gradient descent. This is modeled as a bilevel optimization problem [8] in [20]. However, this is too complex, even impractical, for searching large models (e.g. ResNet and Inception-V3) on large datasets (e.g. ImageNet) without a proxy task. To avoid this, we consider two more efficient optimization approaches. The first, which is used by default, is vanilla end-to-end backpropagation. It treats the architecture $\{\alpha, \beta\}$ and filter $\mathbf{W}$ parameters equally, updating both in a single forward-backward pass. The second is an alternating optimization of two steps: 1) fix $\{\alpha, \beta\}$ and update $\mathbf{W}$; 2) fix $\mathbf{W}$ and update $\{\alpha, \beta\}$. It has twice the complexity of vanilla backpropagation. Our experiments show that these strategies are efficient and effective for MPS.

4.7. Architecture Discretization

Given the optimal architecture parameters $\{\alpha^*, \beta^*\}$, the mixed-precision network must be derived by discretizing the soft selector variables π of (11) into the binary selectors o required by (4). Two strategies are explored.

The first is a "winner-take-all" approach, used by default, in which only the branch with the highest π is selected and the others removed, by defining

$$o_i^* = \begin{cases} 1, & \text{if } i = \arg\max_j \pi_j, \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

This results in a deterministic architecture, which is not affected by the details of the distribution of π, only the relative ranking of its values.

The second is a sampling strategy. Since each π is a categorical distribution, hard selectors o_i can be sampled from a multinomial distribution

$$O \sim \mathbf{Multinomial}(n, \pi). \tag{19}$$

with one trial $n = 1$. This defines a random architecture with expected BitOps as defined in (13). The user can then sample multiple architectures and choose the one with the best trade-off between accuracy and complexity, as in [32]. However, in our experience, the variance of this random architecture is very large, making it very inefficient to find the best network configuration. Alternatively, it is possible to sample from (19) with multiple trials, e.g. $n = 50$, and choose the architecture with the highest counts. This produces architectures closer to that discovered by "winner-take-all", converging to the latter when $n \to \infty$.

5. Experiments

EdMIPS was evaluated with ImageNet [25], top-1 and top-5 classification accuracy. Previous MPS works [32, 31]

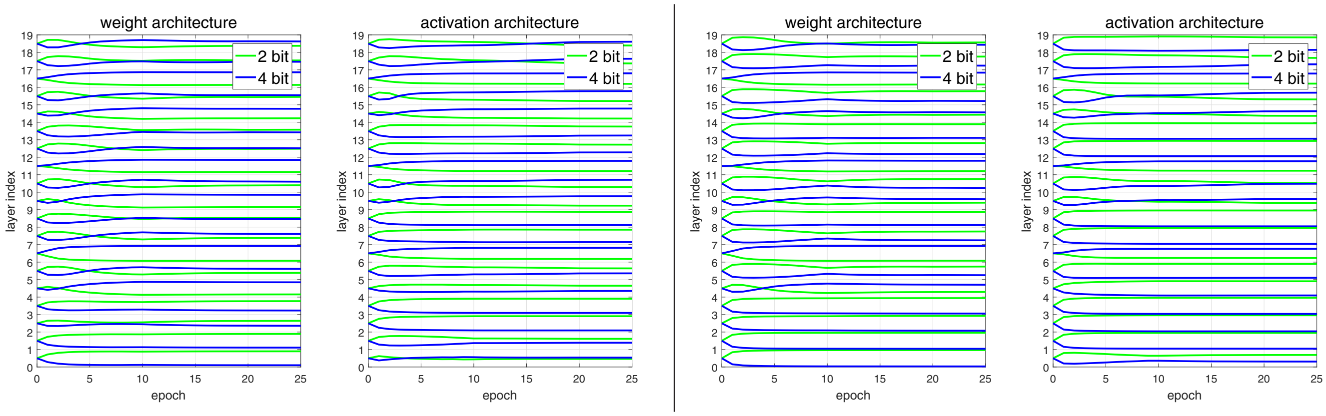


Figure 4. Architecture evolution of ResNet-18 during search. The curves shown per layer index represent the probabilities of the two bit-widths. Layers indexes are as in PyTorch, where 7/12/17 are residual connections. Left: $\eta = 0.001$ in (6). Right: $\eta = 0.002$.

used ResNet-18 or MobileNet [13]. However, these are relatively simple networks, that stack replicas of a single module block, and for which heuristics can be used to hand-craft nearly optimal bit allocation. For example, the depth-wise layers of MobileNet should receive more bits than the point-wise ones. On the other end of the spectrum, GoogLeNet [29] and Inception-V3 [30] are complex combinations of different modules and too complicated to optimize by hand. Since this is the case where MPS is more useful, beyond the small and simple AlexNet and ResNet-18, we have also tested EdMIPS on larger and more complicated ResNet-50, GoogLeNet and Inception-V3. These experiments establish broad baselines for future studies in MPS.

5.1. Implementation Details

All experiments, followed standard ImageNet training on PyTorch[2], with the following exceptions. For simplicity, all auxiliary losses were removed from the GoogLeNet and Inception-V3. EdMIPS used a learning rate of 0.1 for network parameters $\mathbf{W}$, and 0.01 for architecture parameters $\{\alpha, \beta\}$. All network parameters were initialized as usual, and all architecture parameters were set to 0.01, treating all candidates equally. The search model was trained for 25 epochs, with learning rates decayed by 10 time at every 10 epoches. After training the search model, the classification model was derived by the "winner-take-all" strategy of Section 4.7. Next, the classification models were trained for 50 (95) epochs, with learning rate decayed by 10 times at every 15 (30) epochs, to allow exploration (the final comparison with the state-of-the-art). All models were trained from scratch. For network quantization, we followed [4] but enabled scaling in all BatchNorm layers.

5.2. Architecture Evolution

Sometimes, it is mysterious why and how an architecture is found by NAS. Due to the differentiability of EdMIPS, the soft architecture evolves gradually during search, as shown in Figure 4. For clarity, we only show the evolution of a ResNet-18 with search space of $\{2, 4\}$ bit. Some interesting observations follow from the left part of Figure 4. First, the use of (6) with complexity penalty $\eta = 0.001$ is shown to avoid the trivial solution (highest bit-width always selected). Candidates of lower bit-width are selected for many layers. Second, nearly all layers choose 2 bits in the early epochs, where the network parameters are not yet trained and the complexity penalty dominates. As network parameters become stronger, the error penalty is more likely to overcome the complexity penalty, and 4-bit candidates start to be selected. Third, weights and activations can have different optimal bit allocations. For example, 4-bits are selected for the weights of layers 5/6/8, whose activations are assigned 2-bits. Fourth, since the complexity penalty of residual connections (layers 7/12/17) is much smaller than 3×3 convolutions, they are allocated 4-bits early on. Finally, both candidates are equally strong for some layers, e.g. 9/13 for weights and 1 for activations. In these cases, the model is not confident on which candidate to choose.

5.3. Effect of Complexity Constraint

When a stronger complexity constraint is enforced, lower bit-widths are more likely to be chosen. This can be observed by comparing the left ($\eta = 0.001$) and right ($\eta = 0.002$) of Figure 4. For layers preferring lower bit-width, this preference is reinforced by stronger η. For example, in layers 1-4 for weights and 2-6 for activations, EdMIPS converges to lower bit-widths much faster. For layers preferring lower bit-width at the start but switching to more bits at the end, e.g. 5/10/15 for weights and 10/15/16 for activations, more epochs are required to reach the crossing point. The stronger complexity constraint can also change the final decisions. For example, for weights at layers 6/8/11 and activations at layers 18-19, the decisions switch from 4- to 2-bit. These results show that the optimal architecture depends on the complexity constraint.

[2] https://github.com/pytorch/pytorch

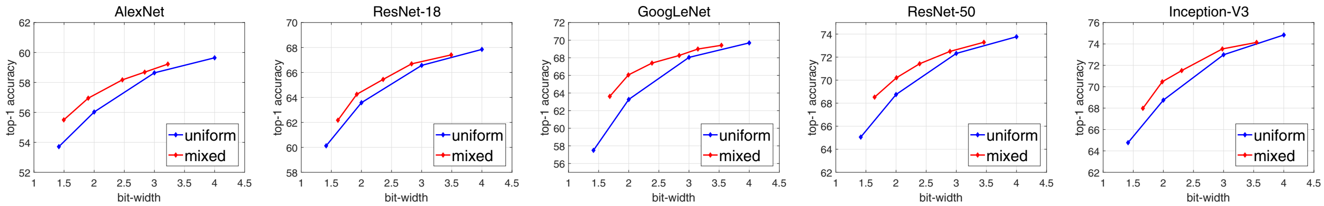


Figure 5. Comparison of the uniform HWGQ-Net and the EdMIPS network. The x-axis, indicating BitOps, is normalized to the scale of bit-width, which is actually in log-scale.

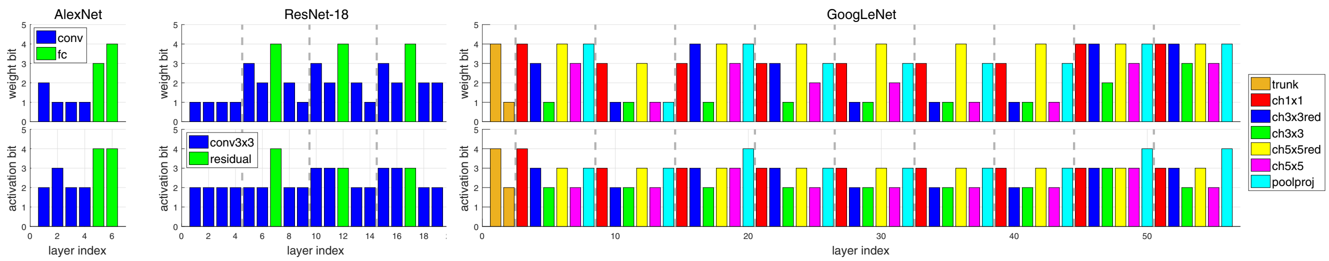


Figure 6. EdMIPS bit allocation for AlexNet, ResNet-18 and GoogLeNet.

5.4. Comparison to Uniform Bit Allocation

Figure 5 shows that EdMIPS networks substantially outperform the uniform HWGQ-Net versions of AlexNet, ResNet-18/50, GoogLeNet, and Inception-V3. Note that the HWGQ-Net has fairly high baselines. Since the learned model is bounded by the weakest (W1A2) and strongest (W4A4) models in the search space, the improvements are small on both ends, but substantial in between. For example, the EdMIPS models improve the uniform 2-bit HWGQ-Net by about 0.9 point for AlexNet, 0.8 point for ResNet-18, 2.8 points for GoogLeNet, 1.5 points for ResNet-50 and 1.7 points for Inception-V3. This is strong evidence for the effectiveness of EdMIPS.

5.5. Learned Optimal Bit Allocation

To understand what EdMIPS learns, we visualize the optimal bit allocation of each layer in Figure 6, for AlexNet, ResNet-18 and GoogLeNet, with roughly the same BitOps as the uniform 2-bit models. On AlexNet, whose FC layers are lighter and closer to the output, they receive higher bit-width. The layers closer to the input, e.g. 1st for weights and 2nd for activations, also receive relatively higher bit-width. On ResNet-18, the cheap residual connections usually receive the largest bit-width. The first block of layers in each stage, e.g. 5-6th, 10-11th, and 15-16th, also has relatively higher bit-width. GoogLeNet is a more complicated network, and so is its bit allocation. The most expensive "ch3x3" layers frequently receive the lowest bit-width, but not always. Higher bit-widths are allocated to "ch3x3" layers closer to the output. On the other hand, "ch5x5red" is very cheap, and usually receives the highest bit-width. Although "ch1x1" is the second most expensive filter, it is frequently allocated higher bit-width. This reflects its highest sensitivity, shown in Figure 3. Finally, "ch3x3red" and "ch5x5" have similar computation, but the former is much more sensitive. Hence, it receives higher bit-width allocations. Note the difficulty of hand-crafting the optimal bit allocation for this complex network.

5.6. Ablation Studies

ResNet-18 is used for ablation experiments. The default EdMIPS model is denoted as "mixed" in Figure 7 (a).

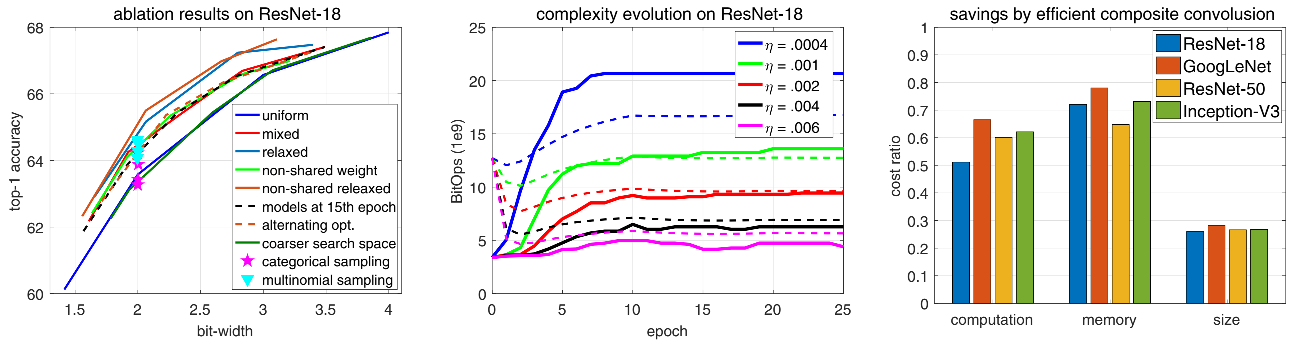
Relaxation: Figure 7 (a) shows that the relaxed models of (10) (trained for 50 epochs), only have slightly higher accuracy than their discretized counterparts, i.e. architecture discretization has little performance cost.

Filter Sharing vs. Non-Sharing: In the default model, all parallel candidates share the weight tensor of (16). Figure 7 (a) shows that this is as effective as learning the individual weight tensors of (15). This is also confirmed by the comparison of their relaxed counterparts, indicating that the unshared parallel branches are somewhat redundant.

Sampling: Four architectures were sampled from the architecture distribution of the model close to 2-bit of Figure 5, using both the categorical and multinomial sampling discussed in Section 4.7, with close BitOps to the uniform 2-bit model. Figure 7 (a) shows that categorical sampling architectures have large accuracy variance, sometimes even underperforming the uniform baseline. Better performance and lower variance are obtained with multinomial sampling.

Learning Strategy: Figure 7 (a) shows that there is little difference in accuracy between the default model, which uses single pass optimization, and a model trained with the alternating optimization of Section 4.6. Since the latter has twice the learning complexity, single pass optimization is used by default on EdMIPS.

Convergence Speed: Figure 7 (b) summarizes the evolu-



(a) (b) (c)

Figure 7. (a) ablation experiments on ResNet-18; (b) complexity evolution on ResNet-18 (dashed lines are expected complexities of (13)); (c) savings by efficient composite convolution of Section 4.4.

tion of network complexity as a function of η, for the five ResNet-18 models of Figure 5. The search usually converges quickly, with minor changes in architecture complexity after the 15th epoch. The accuracy of the corresponding classification models ("models at 15th epoch") is shown in Figure 7 (a). They are comparable to those found after 25 search epochs.

Search Space: Figure 7 (a) shows that no gain occurs over the uniform model when a coarser search space ($\{1, 4\}$ bit for weight and $\{2, 4\}$ bit for activation) is used.

Efficient Composite Convolution: Figure 7 (c) summarizes the practical savings achieved with the efficient composite convolution of Section 4.4, with respect to the vanilla parallel convolutions, in terms of computation, memory and model size, for the search space of (17). The weight sharing of (16) reduces model size by a factor of almost four. Replacing the parallel convolutions results in 30-50% savings in computation and 20-40% in memory. These savings make MPS much more practical. For example, the time needed to search for the ResNet-18 on 2 GPUs decreased from 35 to 18 hours, and the search for the Inception-V3 can be performed with 8 GPUs and 12GB of memory. In result, the complete search (e.g. 25 epochs) only increases by $\sim$45% the training complexity of a uniform low-precision ResNet-18 network. It should be mentioned that these practical savings are nevertheless smaller than theoretically predicted by (14), due to other practical cost bottlenecks, e.g. weight/activation quantization, parallelization efficiency in GPUs, network architecture, etc. Better implementations of EdMIPS could enable further savings.

5.7. Comparison to state-of-the-art

Table 1 compares EdMIPS models to state-of-the-art uniform low-precision networks, including HWGQ-Net [4] and LQ-Net [34]. EdMIPS models, with similar BitOps to the 2-bit uniform models, achieve top performance for all base networks. Compared to the uniform HWGQ-Net, the gains are particularly large (3 points) for the complicated

Table 1. Comparison to the state-of-the-art.

Model bit-width		Ref 32	Full 32	HWGQ 2	LQ [34] 2	EdMIPS $\sim$2
AlexNet	Top-1	57.1	59.2	58.6	57.4	59.1
	Top-5	80.2	81.7	80.9	80.1	81.0
ResNet-18	Top-1	69.6	70.2	65.1	64.9	65.9
	Top-5	89.2	89.5	86.2	85.9	86.5
GoogLeNet	Top-1	73.3	72.7	64.8	-	67.8
	Top-5	91.3	91.0	86.3	-	88.0
ResNet-50	Top-1	76.0	76.2	70.6	71.5	72.1
	Top-5	93.0	93.0	89.8	90.3	90.6
Inception-V3	Top-1	77.5	77.3	71.0	-	72.4
	Top-5	93.6	93.6	89.9	-	90.7

GoogLeNet, a compact model very averse to quantization. On the other hand, for the simplest model (AlexNet), the 2-bit EdMIPS model is already equivalent to the full precision network. These results show that EdMIPS is an effective MPS solution. Since 1) [31] only experiments on MobileNet and uses latency as complexity indicator; 2) [32] mainly focuses on ResNet-18/34 of higher bit-widths, e.g. 4-bit; 3) neither of these works have released the code needed to reproduce their results, we are unable to compare to them. We believe that the release of EdMIPS code and the solid baselines now established by Table 1 will enable more extensive comparisons to future work in MPS.

6. Conclusion

We have proposed EdMIPS, an efficient framework for MPS based on a differentiable architecture. EdMIPS has multiple novel contributions to the MPS problem. It can search large models, e.g. ResNet-50 and Inception-V3, directly on ImageNet, at affordable costs. The learned mixed-precision models of multiple popular networks substantially outperform their uniform low-precision counterparts and establish a solid set of baselines for future MPS research.

Acknowledgment This work was partially funded by NSF awards IIS-1637941, IIS-1924937, and NVIDIA GPU donations.

References

[1] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Fixed point optimization of deep convolutional neural networks for object recognition. In *ICASSP*, pages 1131–1135, 2015. 1, 2

[2] Gabriel Bender, Pieter-Jan Kindermans, Barret Zoph, Vijay Vasudevan, and Quoc V. Le. Understanding and simplifying one-shot architecture search. In *ICML*, pages 549–558, 2018. 1, 2, 4

[3] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. SMASH: one-shot model architecture search through hypernetworks. In *ICLR*, 2018. 1, 2

[4] Zhaowei Cai, Xiaodong He, Jian Sun, and Nuno Vasconcelos. Deep learning with low precision by half-wave gaussian quantization. In *CVPR*, pages 5406–5414, 2017. 1, 2, 3, 5, 6, 8

[5] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, pages 6154–6162, 2018. 1

[6] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(4):834–848, 2018. 1

[7] Jungwook Choi, Zhuo Wang, Swagath Venkataramani, Pierce I-Jen Chuang, Vijayalakshmi Srinivasan, and Kailash Gopalakrishnan. Pact: Parameterized clipping activation for quantized neural networks. *arXiv preprint arXiv:1805.06085*, 2018. 2, 5

[8] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals OR*, 153(1):235–256, 2007. 5

[9] Zhen Dong, Zhewei Yao, Amir Gholami, Michael W. Mahoney, and Kurt Keutzer. HAWQ: hessian aware quantization of neural networks with mixed-precision. In *ICCV*, pages 293–302. IEEE, 2019. 1

[10] Zichao Guo, Xiangyu Zhang, Haoyuan Mu, Wen Heng, Zechun Liu, Yichen Wei, and Jian Sun. Single path one-shot neural architecture search with uniform sampling. *arXiv preprint arXiv:1904.00420*, 2019. 2, 5

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, pages 2980–2988, 2017. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1

[13] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *CoRR*, abs/1704.04861, 2017. 6

[14] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *NIPS*, pages 4107–4115, 2016. 1, 2, 3, 5

[15] Kyuyeon Hwang and Wonyong Sung. Fixed-point feedforward deep neural network design using weights +1, 0, and -1. In *IEEE Workshop on Signal Processing Systems*, pages 174–179, 2014. 2

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1106–1114, 2012. 1

[17] Griffin Lacey, Graham W. Taylor, and Shawki Areibi. Stochastic layer-wise precision in deep neural networks. In *UAI*, pages 663–672, 2018. 1, 2

[18] Darryl Dexu Lin, Sachin S. Talathi, and V. Sreekanth Annapureddy. Fixed point quantization of deep convolutional networks. In *ICML*, pages 2849–2858, 2016. 1, 2

[19] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 936–944, 2017. 1

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018. 1, 2, 4, 5

[21] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Information Theory*, 28(2):129–136, 1982. 3

[22] Joel Max. Quantizing for minimum distortion. *IRE Trans. Information Theory*, 6(1):7–12, 1960. 3

[23] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *ECCV*, pages 525–542, 2016. 1, 2, 3, 5

[24] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 1

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[26] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. 1

[27] Huifang Sun and Yun Q Shi. *Image and video compression for multimedia engineering: Fundamentals, algorithms, and standards*. CRC press, 2008. 3

[28] Wonyong Sung and Ki-Il Kum. Simulation-based word-length optimization method for fixed-point digital signal processing systems. *IEEE Trans. Signal Processing*, 43(12):3087–3090, 1995. 2

[29] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, pages 1–9, 2015. 1, 3, 6

[30] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016. 1, 2, 6

[31] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision. In *CVPR*, pages 8612–8620, 2019. 1, 2, 5, 8

[32] Bichen Wu, Yanghan Wang, Peizhao Zhang, Yuandong Tian, Peter Vajda, and Kurt Keutzer. Mixed precision quantization of convnets via differentiable neural architecture search. *arXiv preprint arXiv:1812.00090*, 2018. 1, 2, 5, 8

[33] Amir Yazdanbakhsh, Ahmed T Elthakeb, Prannoy Pilligundla, FatemehSadat Mireshghallah, and Hadi Esmaeilzadeh. Releq: An automatic reinforcement learning approach for deep quantization of neural networks. *arXiv preprint arXiv:1811.01704*, 2018. 1, 2

[34] Dongqing Zhang, Jiaolong Yang, Dongqiangzi Ye, and Gang Hua. Lq-nets: Learned quantization for highly accurate and compact deep neural networks. In *ECCV*, pages 373–390, 2018. 1, 2, 3, 5, 8

[35] Shuchang Zhou, Zekun Ni, Xinyu Zhou, He Wen, Yuxin Wu, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *CoRR*, abs/1606.06160, 2016. 1, 2, 3, 5

[36] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian Reid. Structured binary neural networks for image recognition. *arXiv preprint arXiv:1909.09934*, 2019. 2

[37] Bohan Zhuang, Chunhua Shen, Mingkui Tan, Lingqiao Liu, and Ian D. Reid. Towards effective low-bitwidth convolutional neural networks. In *CVPR*, pages 7920–7928, 2018. 1, 2, 3, 5

[38] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *CoRR*, abs/1611.01578, 2016. 1, 2, 4

[39] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *CVPR*, pages 8697–8710, 2018. 1, 2