2021 IEEE/CVF International Conference on Computer Vision (ICCV) | 978-1-6654-2812-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ICCV48922.2021.01201

Distilling Optimal Neural Networks: Rapid Search in Diverse Spaces

Bert Moons, Parham Noorzad, Andrii Skliar, Giovanni Mariani,
Dushyant Mehta, Chris Lott and Tijmen Blankevoort
Qualcomm AI Research*
{bmoons,parham,askliar,gmariani,dushmeht,clott,tijmen}@qti.qualcomm.com

Abstract

Current state-of-the-art Neural Architecture Search (NAS) methods neither efficiently scale to multiple hardware platforms, nor handle diverse architectural search-spaces. To remedy this, we present DONNA (Distilling Optimal Neural Network Architectures), a novel pipeline for rapid, scalable and diverse NAS, that scales to many user scenarios. DONNA consists of three phases. First, an accuracy predictor is built using blockwise knowledge distillation from a reference model. This predictor enables searching across diverse networks with varying macro-architectural parameters such as layer types and attention mechanisms, as well as across micro-architectural parameters such as block repeats and expansion rates. Second, a rapid evolutionary search finds a set of pareto-optimal architectures for any scenario using the accuracy predictor and on-device measurements. Third, optimal models are quickly fine-tuned to training-from-scratch accuracy. DONNA is up to 100× faster than MNasNet in finding state-of-the-art architectures on-device. Classifying ImageNet, DONNA architectures are 20% faster than EfficientNet-B0 and MobileNetV2 on a Nvidia V100 GPU and 10% faster with 0.5% higher accuracy than MobileNetV2-1.4x on a Samsung S20 smartphone. In addition to NAS, DONNA is used for search-space extension and exploration, as well as hardware-aware model compression.

1. Introduction

Although convolutional neural networks (CNN) have achieved state-of-the-art performance for a wide range of vision tasks, they do not always execute efficiently on hardware platforms like desktop GPUs or mobile DSPs and NPUs. To alleviate this issue, CNNs are specifically optimized to minimize latency and energy consumption for on-device performance. However, the optimal CNN architecture can vary significantly between different platforms. Even on a single platform, their efficiency can change with different operating conditions or driver versions. To solve this problem, low-cost methods for automated hardware-aware neural architecture search (NAS) are required.

Current NAS algorithms, however, suffer from several limitations. First, many optimization algorithms [32, 12, 31, 20] target only a single *deployment scenario*: a hardware-agnostic complexity metric, a hardware platform, or different latency, energy, or accuracy requirements. This means the search has to be repeated whenever any part of that scenario changes. Second, many methods cannot search in truly diverse search spaces, with different types of convolutional kernels, activation functions and attention mechanisms. Current methods either search through large and diverse spaces at a prohibitively expensive search cost [32, 12], or limit their applicability by trading search time for a more constrained and less diverse search [3, 31, 33, 41, 23, 22]. Most of such speedups in NAS come from a reliance on weight sharing mechanisms, which require all architectures in the search space to be structurally similar. Thus, these works typically only search among *micro-architectural* choices such as kernel sizes, expansion rates, and block repeats and not among *macro-architectural* choices of layer types, attention mechanisms and activation functions. As such, they rely on prior expensive methods such as [32, 12] for an optimal choice of macro-architecture.

We present DONNA (Distilling Optimal Neural Network Architectures), a method that addresses both issues: it scales to multiple deployment scenarios with low additional cost, and performs rapid NAS in diverse search spaces. The method starts with a trained reference model. The first issue is resolved by splitting NAS into a scenario-agnostic training phase, and a scenario-aware search phase that requires only limited training, as depicted in Figure 1. After an accuracy predictor is built in the training phase, the search is executed quickly for each new deployment scenario, typically in the time-frame of hours, and only requiring minimal fine-tuning to finalize optimal models. Second, DONNA considers diverse *macro-architectural* choices in addition to *micro-architectural* choices, by creating this ac-

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

978-1-6654-2812-5/21/$31.00 ©2021 IEEE
DOI 10.1109/ICCV48922.2021.01201

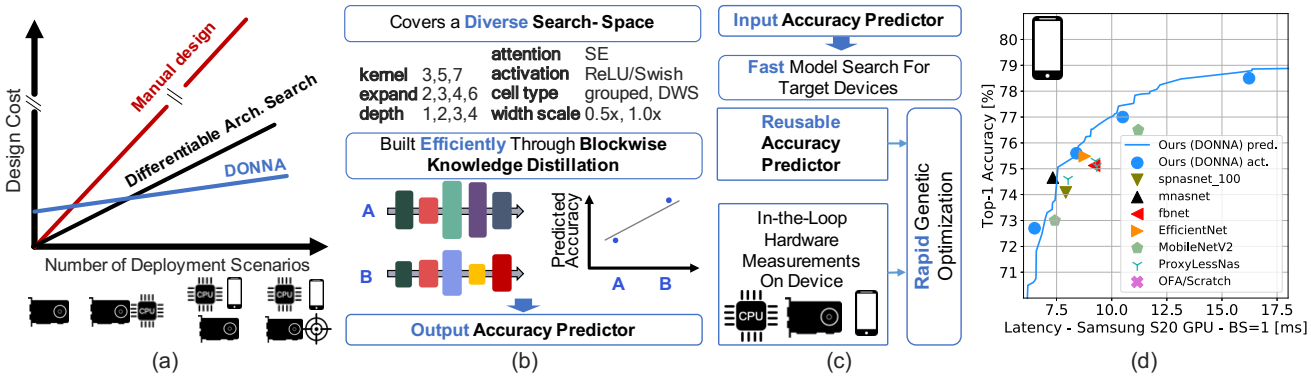

Figure 1. Neural networks are deployed in many scenarios, on various hardware platforms with varying power modes and driver software, with different speed and accuracy requirements. DONNA scales gracefully towards NAS for many of such scenarios, contrary to most prior approaches where NAS is repeated for each of them (a). This is achieved by splitting NAS into a scenario-agnostic training phase building an accuracy predictor through blockwise knowledge distillation (b) and a rapid scenario-aware search phase using this predictor and hardware measurements (c). This yields a Pareto-front of models on-device, shown here for a Samsung S20 GPU on ImageNet [8] (d).

curacy predictor through Blockwise Knowledge Distillation (BKD) [18], see Figure 3. This approach imposes little constraints on the macro- and micro-architectures under consideration, allowing a vast, diverse, and extensible search space. The DONNA pipeline yields state of the art network architectures, as illustrated for a Samsung S20 GPU in Figure 1(d). Finally, we use DONNA for rapid search space extension and exploration, and on-device model compression. This is possible as the DONNA accuracy predictor generalizes to architectures outside the original search space.

2. Related Work

Over time, methods in the NAS literature have evolved from prohibitively expensive but holistic and diverse search methods [42, 43, 32] to lower cost approaches that search in more restrictive non-diverse search spaces [3, 31]. This work, DONNA, aims at benefiting from the best of both worlds: rapid search in diverse spaces. We refer the interested reader to the existing dedicated survey of Elsken et al. [10] for a broader discussion of the NAS literature.

Early approaches to NAS rely on reinforcement learning [42, 43, 32] or evolutionary optimization [29]. These methods allow for diverse search spaces, but at infeasibly high costs due to the requirement to train thousands of models for a number of epochs throughout the search. MNasNet [32] for example uses up to 40,000 epochs in a single search. This process can be sped up by using weight sharing among different models, as in ENAS [28]. However, this comes at the cost of a less diverse search space, as the subsampled models have to be similar for the weights to be shareable.

In another line of work, differentiable architecture search methods such as DARTS [20], FBNet [38], FBNetV2 [35], ProxylessNAS [4], AtomNAS [24] and Single-Path NAS [31] simultaneously optimize the weights of a large supernet and its architectural parameters. This poses several impediments to scalable and scenario-aware NAS in diverse search spaces. First, in most of these works, different cell choices have to be available to the algorithm, ultimately limiting the space's size and diversity. While several works address this problem either by trading off the number of architecture parameters against the number of weights that are in GPU memory at a given time [5], by updating only a subset of the weights during the search [40], or by exploiting more granular forms of weight-sharing [31], the fundamental problem remains when new operations are introduced. Second, although differentiable search methods speed up a single search iteration, the search must be repeated for every scenario due to their coupling of accuracy and complexity. Differentiable methods also require differentiable cost models. Typically these models use the sum of layer latencies as a proxy for the network latency, which can be inaccurate. This is especially the case in emerging depth-first processors [11], where intermediate results are stored in the local memory, making full-graph latency depend on layer sequences rather than on individual layers.

To improve the scaling performance of NAS across different scenarios, it is critical to decouple the accuracy prediction of a model from the complexity objective. In Once-for-All (OFA) [3] and [22], a large weight-sharing supernet is trained using progressive shrinking. This process allows the sampling of smaller subnets from the trained supernet that perform comparably with models that have been trained from scratch. A large number of networks can then be sampled to build an accuracy predictor for this search space, which in turn can be used in a scenario-aware evolutionary search, as in Figure 1(c). Although similar to DONNA in this approach, OFA [3] has several disadvantages. First, its search space's diversity is limited due to its reliance on progressive shrinking and weight sharing, which requires a fixed macro-architecture in terms of layer types, attention,

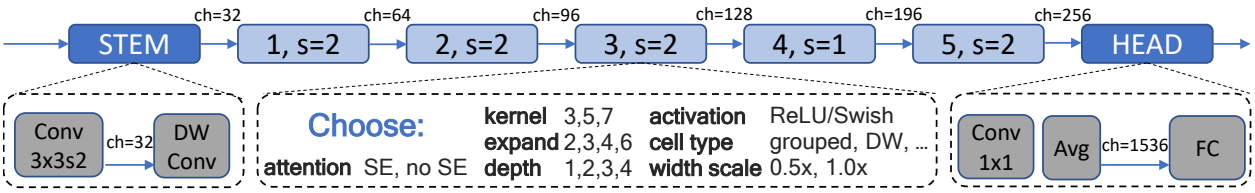

Figure 2. DONNA splits a model in a stem, head and N blocks. The search space is defined over the N blocks with varying kernel size, expand, depth, activation, cell type, attention and width scale factors. Block strides are kept constant.

activations, and channel widths. Furthermore, progressive shrinking can only be parallelized in the batch dimension, limiting the maximum number of GPUs that can process in parallel. DONNA does not suffer from these constraints.

Similarly, Blockwisely-Supervised NAS (DNA) [18], splits NAS into two phases: the creation of a ranking model for a search space and a targeted search to find the highest-ranked models at a given constraint. To build this ranking model, DNA uses blockwise knowledge distillation (BKD) to build a relative ranking of all possible networks in a given search space. The best networks are then trained and verified. It is crucial to note that it is BKD that enables the diverse search for optimal attention mechanisms, activation functions, and channel scaling. However, DNA has three disadvantages: (1) the ranking model fails when ranking large and diverse search spaces (Section 3.2), (2) the ranking only holds within a search space and does not allow the comparison of different spaces, and (3) because of the reliance on training subsampled architectures from scratch, the method is not competitive in terms of search time. This work, DONNA, addresses all these issues. In summary, DONNA differs from prior work on these key aspects:

1. Unlike OFA [3], DONNA enables hardware-aware search in *diverse search spaces;* differentiable and RL-/evolutionary-based methods can do this too, but using much more memory or training time, respectively.

2. DONNA *scales to multiple accuracy/latency targets*, requiring only marginal cost for every new target. This is in contrast with differentiable or RL-/evolutionary-based methods, where the search has to be repeated for every new target.

3. DONNA uses *a novel accuracy predictor* which correlates better with training-from-scratch accuracy than prior work like DNA [18] (See Figure 4).

4. Furthermore, the DONNA accuracy predictor *generalizes to unseen search spaces* due to its reliance on block *quality metrics*, not on the network configuration (See Figure 7).

5. DONNA relies on a *fast finetuning* method that achieves the same accuracy as training-from-scratch while being $9\times$ faster, reducing the training time for found architectures compared to DNA [18].

3. Distilling Optimal Neural Networks

Starting with a trained reference model, DONNA is a three step pipeline for NAS. For a given search space (Section 3.1), we first build a scenario-agnostic accuracy predictor using Blockwise Knowledge Distillation (BKD) (Section 3.2). This amounts to a one-time cost. Second, a rapid scenario-aware evolutionary search phase finds the Pareto-optimal network architectures for any specific scenario (Section 3.3). Third, the predicted Pareto-optimal architectures can be quickly finetuned up to full accuracy for deployment (Section 3.4).

3.1. Search Space Structure

Figure 2 illustrates the block-level architecture of our search spaces and some parameters that can be varied within it. This search space is comprised of a stem, head, and N variable blocks, each with a fixed stride. The choice of stem, head and the stride pattern depends on the choice of the reference model. The blocks used here are comprised of repeated layers, linked together by feedforward and residual connections. The blocks in the search space are denoted $B_{n,m}$, where $B_{n,m}$ is the m^{th} potential replacement out of M choices for block B_n in the reference model. These blocks can be of any style of neural architecture (See Appendix C for Vision Transformers [9]), with very few structural limitations; only the spatial dimensions of the input and output tensors of $B_{n,m}$ need to match those of the reference model, which allows for diverse search. Throughout the text and in Appendix A, other reference models based on MobileNetV3 [12] and EfficientNet [33] are discussed.

3.2. Building a Model Accuracy Predictor

3.2.1 Blockwise Knowledge Distillation

We discuss Blockwise Knowledge Distillation (BKD) as the first step in building an accuracy predictor for our search space, see Figure 3(a). BKD yields a *Block Library* of pretrained weights and quality metrics for each of the replacement blocks $B_{n,m}$. This is later used for fast finetuning (Section 3.4) and to fit the accuracy predictor (Section 3.2.2). To build this library, each block $B_{n,m}$ is trained independently as a student using the pretrained reference block B_n as a teacher. The errors between the teacher's output feature map Y_n and the student's output feature map

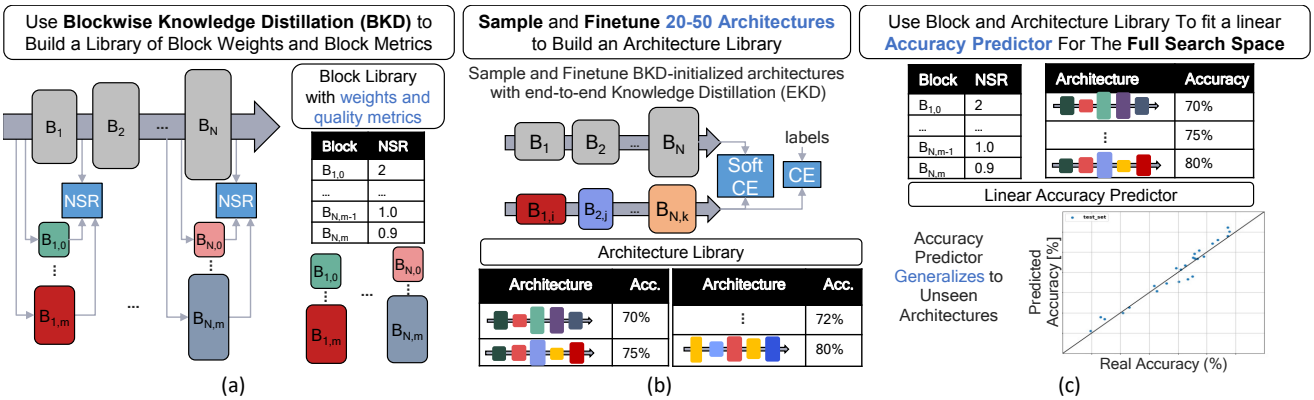

Figure 3. An accuracy predictor is built in three steps. (a) Blockwise knowledge distillation (BKD) is executed to build a library of block-quality metrics and pretrained weights. (b) A set of full-model architectures is sampled from the search space and finetuned using the BKD initialization. (c) These results are used as targets to fit a linear accuracy predictor.

$\bar{Y}_{n,m}$ are used in this process. Formally, this is done by minimizing the per-channel noise-to-signal-power ratio (NSR):

$$\mathcal{L}(W_{n,m}; Y_{n-1}, Y_n) = \frac{1}{C} \sum_{c=0}^{C} \frac{Y_{n,c} - \bar{Y}_{n,m,c}{}^2}{\sigma_{n,c}^2} \quad (1)$$

Here, C is the number of channels in a feature map, $W_{n,m}$ are the weights of block $B_{n,m}$, Y_n is the target output feature map of B_n, $\bar{Y}_{n,m}$ is the output of block $B_{n,m}$ and $\sigma_{n,c}^2$ is the variance of $Y_{n,c}$. This metric is closely related to Mean-Square-Error (MSE) on the feature maps, which [25] shows to be correlated to the task loss.

Essentially, the blocks $B_{n,m}$ are trained to closely replicate the teacher's non-linear function $Y_n = B_n(Y_{n-1})$. Intuitively, larger, more accurate blocks with a larger "modeling capacity" or "expressivity" replicate this function more closely than smaller, less accurate blocks. On ImageNet [8] such knowledge distillation requires only a *single* epoch of training for effective results. After training each block, the resulting NSR metric is added to the Block library as a *quality metric* of the block $B_{n,m}$. Note that the total number of trainable blocks $B_{n,m}$ grows linearly as $N \times M$, whereas the overall search space grows exponentially as M^N, making the method scale well even for large search-spaces.

3.2.2 Linear Accuracy Predictor

The key insight behind DONNA is that block-level quality metrics derived through BKD (e.g., per-block NSR) can be used to predict the accuracy of all architectures sampled from the search space. We later show this metric even works for architectures outside of the search space (Section 4.1.2).

To create an accuracy predictor, we build an *Architecture Library* of trained models sampled from the search space, see Figure 3(b). These models can be trained from scratch or finetuned quickly using weight initialization from BKD (Section 3.4). Subsequently, we fit a linear regression

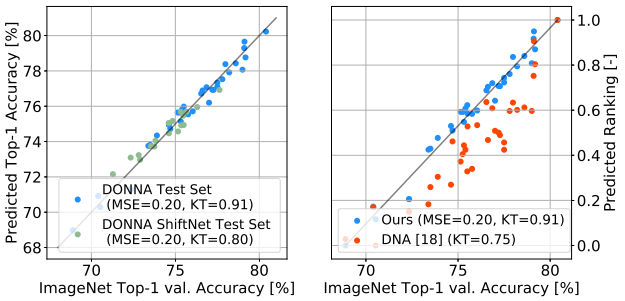

Figure 4. The linear accuracy predictor generalizes to a test-set of unseen models (left), and is a better ranking predictor than DNA [18] (right) on the same set: Kendall-Tau [16] of 0.91 in this work versus 0.75 for DNA.

model, typically using second-order terms, to predict the full search space's accuracy using the quality metrics stored in the Block Library as features and the accuracy from the Architecture Library as targets. Figure 4(left) shows that the linear predictor fits well with a test-set of network architectures trained on ImageNet [8] in the DONNA space (MSE=0.2, KT [16]=0.91). This predictor can be understood as a sensitivity model that indicates which blocks should be large, and which ones can be small, to build networks with high accuracy. Appendix A.4.2 discusses the effectiveness of different derived quality metrics on the quality of the accuracy prediction.

This process is now compared to DNA [18], where BKD is used to build a ranking-model rather than an accuracy model. DNA [18] ranks subsampled architectures i as:

$$R_i = \sum_{n=0}^{N} \frac{Y_n - \bar{Y}_{n,m_i 1}}{\sigma_n} \quad (2)$$

which is sub-optimal due to two reasons. First, a ranking model only ranks models within the same search space and does not allow comparing performance of different search

spaces. Second, the simple sum of quality metrics does not take the potentially different noise-sensitivity of blocks into account, for which a weighted sensitivity model is required. The DONNA predictor takes on both roles. Figure 4(right) illustrates the performance of the linear predictor for the DONNA search space and compares the quality of its ranking to DNA [18]. Note that the quality of the DONNA predictor increases over time, as whenever Pareto-optimal networks are finetuned, they can be added to the Architecture Library, and the predictor can be fitted again.

3.3. Evolutionary Search

Given the accuracy model and the block library, the NSGA-II [7, 1] evolutionary algorithm is executed to find Pareto-optimal architectures that maximize model accuracy and minimize a target cost function, see Figure 1(c). The cost function can be scenario-agnostic, such as the number of operations or the number of parameters in the network, or scenario-aware, such as on-device latency, throughput, or energy. In this work, full-network latency is considered as a cost function by using direct hardware measurements in the optimization loop. At the end of this process, the Pareto-optimal models yielded by the NSGA-II are finetuned to obtain the final models (Section 3.4).

3.4. Finetuning Architectures

Full architectures sampled from the search space can be quickly finetuned to match the from-scratch training accuracy by initializing them with weights from the BKD process (Section 3.2.1). Finetuning is further sped up by using end-to-end knowledge distillation (EKD) using the reference model as a teacher, see Figure 3(b). In Appendix A.5, we show such models can be finetuned up to state-of-the-art accuracy in less than 50 epochs. This is a $9\times$ speedup compared to the state-of-the-art 450 epochs required in [37] for training EfficientNet-style networks from scratch. This rapid training scheme is crucial to the overall efficiency of DONNA, since we use it for both, generating training targets for the linear accuracy predictor in Section 3.2, as well as to finetune and verify Pareto-optimal architectures.

4. Experiments

This section discusses three use-cases of DONNA: scenario-aware neural architecture search (Section 4.1.1), search-space extrapolation and design (Section 4.1.2), and model compression (Section 4.1.3). We also show that DONNA can be directly applied to object detection on MS-COCO [19] and that architectures found by DONNA transfer to optimal detection backbones (Section 4.2). DONNA is compared to random search in Appendix E.

4.1. ImageNet Classification

We present experiments for different search spaces for ImageNet classification: DONNA, EfficientNet-Compression and MobileNetV3 ($1.0\times$, $1.2\times$). The latter two search spaces are *blockwise* versions of the spaces considered by OFA [2]; that is, parameters such as expansion ratio and kernel size are modified on the block level rather than the layer level, rendering the overall search space coarser than that of OFA. Selected results for these spaces are discussed in this section, more extensive results can be found in Appendix A.6. We first show that networks found by DONNA in the DONNA search space outperform the state-of-the-art (Figure 5). For example, DONNA is up to 2.4% more accurate on ImageNet [8] validation compared to OFA[3] trained from scratch with the same amount of parameters. At the same time, DONNA finds models outperforming DNA [18] up to 1.5% on a V100 GPU at the same latency and MobileNetV2 ($1.4\times$) by 10% at 0.5% higher accuracy on the Samsung S20 GPU. We also show that MobileNetV3-style networks found by DONNA achieve the same quality of models compared to Mnasnet [32] and OFA [3] when optimizing for the same metric (See Fig. 6 and Tab. 2). All experiments are for ImageNet [8] images with 224×224 input resolution. Training hyperparameters are discussed in Appendix A.1.

4.1.1 NAS for DONNA on ImageNet

DONNA is used for *scenario-aware Neural Architecture Search* on ImageNet [8], quickly finding state-of-the-art models for a variety of deployment scenarios, see Figure 5.

As shown in Figure 2, all 5 blocks B_n in the DONNA space can be replaced by a choice out of $M = 384$ options: k $\in$ {3,5,7}; expand $\in$ {2,3,4,6}; depth $\in$ {1,2,3,4}; activation/attention $\in$ {ReLU/None, Swish[12]/SE[13]}; layer-type $\in$ {grouped, depthwise inverted residual bottleneck}; and channel-scaling $\in \{0.5\times, 1.0\times\}$. The search-space can be expanded or arbitrarily constrained to known efficient architectures for a device. Each of these $5 \times 384 = 1920$ alternative blocks is trained using BKD to complete the Block Library. Once the Block Library is trained, we use the BKD-based ranking metric from DNA[18] to sample a set of architectures uniformly spread over the ranking space. For the DONNA search space, we finally finetune the sampled networks for 50 epochs starting from the BKD initialization, building an Architecture Library with accuracy targets used to fit the linear accuracy predictor. Typically, 20-30 target networks need to be finetuned to yield good results, see Appendix A.4.

In total, including the training of a reference model (450 epochs), $450 + 1920 + 30 \times 50 = 3870$ epochs of training are required to build the accuracy predictor. This is less than $10\times$ the cost of training a single network from

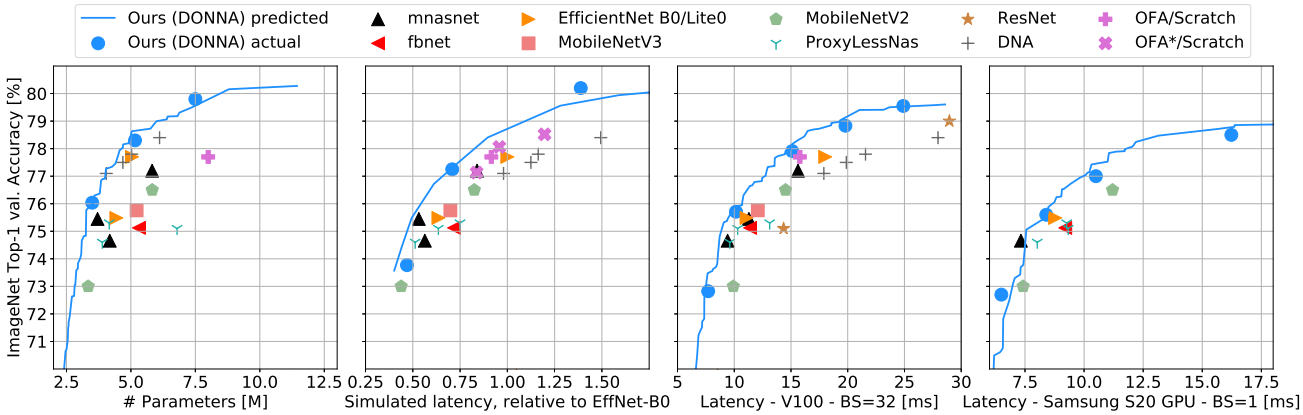

Figure 5. The predicted Pareto-optimal front and models found by DONNA in the DONNA search space. Results are shown targeting the number of operations (left), the number of parameters (mid left), latency on a Nvidia V100 GPU (mid right) and latency on a simulator targeting tensor compute units in a mobile SoC (right). The trend line indicates predicted accuracy, whereas the dots are sampled from the trend line and finetuned up to the level of from-scratch accuracy. OFA*/Scratch results are our own search results using the framework in [2] for 224×224 images, where the best models are retrained from scratch with DONNA hyperparameters for fair comparison.

Table 1. Comparing the cost of NAS methods, assuming 10 trained architectures per deployment scenario. DONNA can search in a diverse space similar to MNasNet [32] at a $100\times$ lower search-cost.

Method	Granularity	Macro-Diversity	Search-cost 1 scenario [epochs]	Cost / Scenario 4 scenarios [epochs]	Cost / Scenario ∞ scenarios [epochs]
OFA [3]	layer-level	fixed	$1200{+}10{\times}[25-75]$	$550-1050$	$250-750$
NSGANetV2 [22]	layer-level	fixed	$1200{+}10{\times}[25-75]$	$550-1050$	$250-750$
DNA [18]	layer-level	fixed	$770{+}10{\times}450$	4700	4500
MNasNet [32]	block-level	variable	$40000{+}10{\times}450$	44500	44500
This work	block-level	variable	$4000+10{\times}50$	1500	500

scratch to model the accuracy of more than 8 trillion architectures. Subsequently, any architecture can be selected and trained to full accuracy in 50 epochs, starting from the BKD initialization. Similarly, as further discussed in Appendix A.4, an accuracy model for MobileNetV3 ($1.2\times$) and EfficientNet-Compressed costs $450 + 135 + 20 \times 50 = 1585$ epochs, roughly the same as training 4 models from scratch. Although this is a higher cost than OFA [3], it covers a much more diverse search space. OFA requires an equivalent, accounting for dynamic batch sizes [2], of $180 + 125 + 2 \times 150 + 4 \times 150 = 1205$ epochs of progressive shrinking with backpropagation on a large supernet. BKDNAS [18] requires only $450 + 16 \times 20 = 770$ epochs to build its ranking model, but 450 epochs to train models from scratch. Other methods like MnasNet [32] can handle a similar diversity as DONNA, but typically require an order of magnitude longer search time (40000 epochs) *for every deployment scenario*. DONNA offers MNasNet-level diversity at a 2 orders of magnitude lower search cost. On top of that, BKD epochs are significantly faster than epochs on a full network, as BKD requires only partial computation of the reference model and backpropagation on a single block $B_{n,m}$. Moreover, and in contrast to OFA, all blocks $B_{n,m}$ can be trained in parallel since they are completely independent of each other. Table 1 quantifies the differences in search-time between these approaches.

With the accuracy predictor in place, Pareto-optimal DONNA models are found for several targets. Figure 5 shows DONNA finds networks that outperform the state of the art in terms of the number of parameters, on a simulator targeting tensor compute units in a mobile SoC, on a NVIDIA V100 GPU and on the Samsung S20 GPU. Every predicted Pareto-optimal front is generated using an evolutionary search with NSGA-II [7, 1] on a population of 100 architectures until convergence. Where applicable, full-architecture hardware measurements are used in the evolutionary loop. Details on measurements and baseline accuracy are given in Appendix A.3.

Similarly, Tab. 2 and Fig. 6 show that DONNA finds models that are on-par with architectures found by other state-of-the-art methods such as MnasNet [32] and OFA [3] in the same spaces. Tab. 2 shows DONNA finds models in the MobileNetV3 ($1.0\times$) space that are on par with MobileNetV3 [12] in terms of number of operations, although [12] is found using expensive MnasNet [32]. Fig. 6 shows the same for networks found through DONNA in the MobileNetV3 ($1.2\times$) search space, by comparing them to models found through OFA [3] optimized for the same

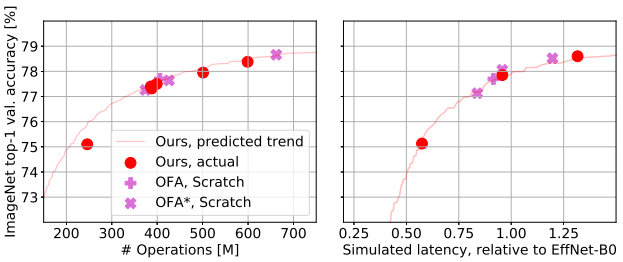

Figure 6. DONNA-NAS finds models that are on-par with models found by OFA [3] in the MobileNetV3 (1.2×) search-space. Models are identically trained for fair comparison. OFA* models are found by us using [2] and trained from scratch.

Table 2. DONNA finds similar models to MobileNetV3 [12] in the MobileNetV3 (1.0×) space.

Network	Number of Operations [M]	ImageNet val top-1 [%]
MobileNetV3 [12]	232	75.77@600[37]
Ours (MobNetV3 1.0×)	242	75.75@50

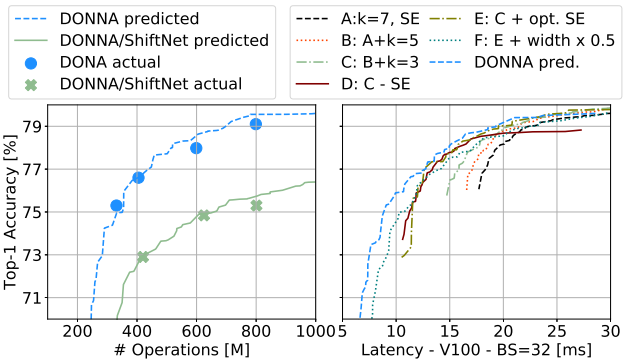

Figure 7. (left) An accuracy predictor for DONNA generalizes to an unseen space with ShiftNets [39], without using ShiftNets to train the predictor. (right) Rapid, model-driven exploration of models within the original DONNA search-space on a V100 GPU. The figure illustrates the necessity of a diverse search space, achieving up to 25% latency gains when attention can be chosen optimally (line E vs C).

complexity metric and trained with the same hyperparameters. More results for other search spaces are shown in Figure 11 in Appendix A.6. We also visualize Pareto-optimal DONNA models for different platforms in Appendix F.

4.1.2 Search-Space Extension and Exploration

The DONNA approach can also be used for *rapid search space extension and exploration*. Using DONNA, a designer can quickly determine whether the search space should be extended or constrained for optimal performance.

Such *extension* is possible because the DONNA accuracy predictor generalizes to previously unseen architectures, without having to extend the Architecture Library. This is illustrated in Fig. 4(left), showing the DONNA predictor achieves good quality, in line with the original test set, on a ShiftNet-based test set of architectures. Figure 7(left) further illustrates this extrapolation works by showing the confirmed results of a search for the ShiftNet space. Note how the trendline predicts the performance of full Pareto optimal ShiftNets even though the predictor is created without any ShiftNet data. Here, ShiftNets are our implementation, with learned shifts per group of 32 channels as depthwise-separable replacement. These generalization capabilities are obtained because the predictor only uses quality metrics as an input without requiring any structural information about the replacement block. This feature is a major advantage of DONNA compared to OFA [3] and other methods where the predictor cannot automatically generalize to completely different layer-types, or to blocks of the same layer-type with parameters (expansion rate, kernel size, depth, ...) outside of the original search space. Appendix D illustrates such extension can also be used to model accuracy of lower precision quantized networks.

This prototyping capability is also showcased for the DONNA search space on a V100 GPU in Figure 7(right). Here we interpolate, using the original accuracy predictor for *exploration*. In doing this, Fig. 7 shows search-space diversity is crucial to achieve good performance. Especially the impact of optimally adding SE-attention [13] is very large, predicting a 25% speedup at 76% accuracy (line C vs D), or a 1% accuracy boost at 26ms (line E vs D). Every plotted line in Figure 7 (right) is a predicted Pareto-optimal. A baseline (A) considers SE/Swish in every block and k $\in$ {7}, expand $\in$ {3,4,6} and depth $\in$ {2,3,4}. Other lines show results for search spaces built starting from (A), e.g. (B) considers k $\in$ {5,7}, (C) k $\in$ {3,5,7}, (D) removes SE/Swish, (E) allows choosing optimal placement of SE/Swish, (F) adds a channel-width multiplier.

4.1.3 Model Compression

DONNA is also used for *hardware-aware compression of existing neural architectures* into faster, more efficient versions. DONNA can do compression not just in terms of the number of operations, as is common in literature, but also for different devices. This is useful for a designer who has prototyped a network for their application and wants to run it efficiently on many different devices with various hardware and software constraints. Figure 8 shows how EfficientNet-B0 can be compressed into networks that are 10% faster than MnasNet [32] on the Samsung S20 GPU.

In the DONNA compression pipeline, the EfficientNet search space splits EfficientNet-B0 into 5 blocks and uses it as the reference model. Every replacement block $B_{n,m}$ considered in compression is smaller than the corresponding reference block. 1135 epochs of training are spent in total to build an accuracy predictor: 135 blocks are trained

using BKD, and 20 architectures are trained for 50 epochs as prediction targets, a cost equivalent to the resources needed for training 3 networks from scratch. Figure 8 shows DONNA finds a set of smaller, Pareto optimal versions of EfficientNet-B0 both in the number of operations and on-device. These are on-par with MobileNetV3 [12] in the number of operations and 10% faster than MnasNet [32] on device. For Samsung S20, the accuracy predictor is calibrated, as these models have no SE and Swish in the head and stem as in the EfficientNet-B0 reference.

Similarly, DONNA can be used to optimally compress Vision Transformers (ViT [9]), see Appendix C.

4.2. Object Detection on MS-COCO

The DONNA architectures transfer to other tasks such as object detection on MS COCO [19]. To this end, we use the EfficientDet-D0 [34] detection architecture, replacing its backbone with networks optimized through the DONNA pipeline. For training, we use the hyperparameters given in [36]. The EfficientDet-D0 initialization comes from [37].

Figure 9 shows the results of multiple of such searches. First, we optimize backbones on ImageNet in the MobileNetV3 (1.2×) and DONNA spaces (ours-224), targetting both the number of operations (left) and latency on a simulator targeting tensor compute units. In this case, the input resolution is fixed to 224×224. The backbones are first finetuned on ImageNet and then transferred to MS-COCO. Second, we apply the DONNA pipeline directly on the full DONNA-det0 architecture, building an accuracy predictor for MS-COCO. We optimize only the backbone and keep the BiFPN head fixed (Ours-COCO-512). In this case, the resulting networks are directly finetuned on MS-COCO, following the standard DONNA-flow. For OFA [3], we consider two sets of models. The first set consists of models optimized for the number of operations (FLOP) with varying input resolution coming directly from the OFA repository [2]. The second set of models, which we identify by 'OFA-224', are obtained by us with the same tools [2], but with the input resolution fixed to 224×224. This makes the OFA-224 search space the same as our MobileNetV3 (1.2×) up to the layerwise-vs-blockwise distinction. In the first experiment, we initialize the OFA backbone with weights from progressive shrinking released in [2]. In the second experiment, we initialize the OFA backbone with from-scratch trained weights on ImageNet using hyperparameters from [37]. After such initialization, the networks are transferred to object detection for comparison. The comparison of the two experiments shows the benefit of OFA-style training is limited after transfer to a downstream task (See Fig. 9.) The gap between OFA-style training and training from scratch, which is up to 1.4% top-1 on ImageNet, decreases to 0.2% mAP on COCO, reducing its importance. We discuss this point further in Appendix B.

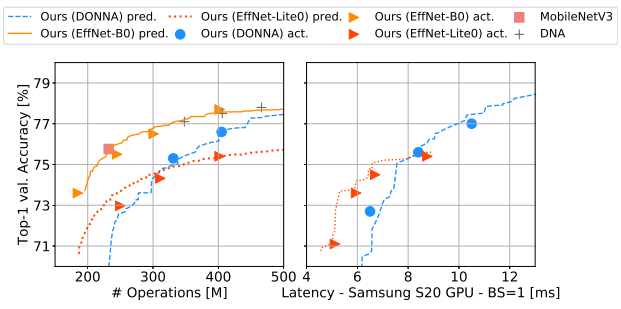

Figure 8. Compressing EfficientNet-B0 for two targets.

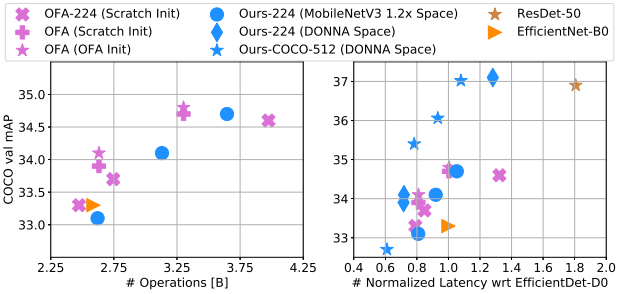

Figure 9. Object detection performance of DONNA backbones, either searched on ImageNet and transferred to COCO (Ours-224), or searched directly on MS COCO (Ours-COCO-512). In the DONNA search space, our solution has up to 2.4% higher mAP at the same latency as the OFA models.

In comparing with DONNA models, we make three key observations. First, models transferred after a search using DONNA are on-par or better than OFA-224 models for both operations and latency. Second, models transferred from the DONNA space outperform OFA models up to 2.4% mAP on the validation set in latency. Third, best results are achieved when applying DONNA directly to MS COCO.

5. Conclusion

In this work, we present DONNA, a novel approach for rapid scenario-aware NAS in diverse search spaces. Through the use of a model accuracy predictor, built through knowledge distillation, DONNA finds state-of-the-art networks for a variety of deployment scenarios: in terms of number of parameters and operations, and in terms of latency on Samsung S20 and the Nvidia V100 GPU. In ImageNet classification, architectures found by DONNA are 20% faster than EfficientNet-B0 and MobileNetV2 on V100 at similar accuracy and 10% faster with 0.5% higher accuracy than MobileNetV2-1.4x on a Samsung S20 smartphone. In object detection, DONNA finds networks with up to 2.4% higher mAP at the same latency compared to OFA. Furthermore, this pipeline can be used for quick search space extensions (e.g. adding ShiftNets) and exploration, as well as for on-device network compression.

References

[1] J. Blank and K. Deb. Pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020. 5, 6

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. once-for-all. https://github.com/mit-han-lab/once-for-all, 2020. 5, 6, 7, 8, 11, 12, 14, 15

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *International Conference on Learning Representations (ICLR)*, 2020. 1, 2, 3, 5, 6, 7, 8, 14

[4] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. *International Conference on Learning Representations (ICLR)*, 2019. 2, 11, 12

[5] Xin Chen, Lingxi Xie, Jun Wu, and Qi Tian. Progressive DARTS: Bridging the optimization gap for nas in the wild. *International Conference on Computer Vision (ICCV)*, 2019. 2

[6] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. RandAugment: Practical automated data augmentation with a reduced search space. *International Conference on Computer Vision (ICCV) Workshop*, 2020. 11

[7] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002. 5, 6, 16

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. 2, 4, 5, 14, 15

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16× 16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 3, 8, 14

[10] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20:1–21, 2019. 2

[11] K. Goetschalckx and M. Verhelst. Breaking high-resolution cnn bandwidth barriers with enhanced depth-first execution. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems (JETCAS)*, 9(2):323–331, 2019. 2

[12] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *International Conference on Computer Vision (ICCV)*, 2019. 1, 3, 5, 6, 7, 8, 13

[13] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 5, 7, 16

[14] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Weinberger. Deep networks with stochastic depth. *European Conference on Computer Vision (ECCV)*, 2016. 11

[15] Evan J Hughes. Multi-objective equivalent random search. In *Parallel Problem Solving from Nature-PPSN IX*, pages 463–472. Springer, 2006. 16

[16] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938. 4, 11, 12, 13, 16

[17] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 11

[18] Changlin Li, Jiefeng Peng, Liuchun Yuan, Guangrun Wang, Xiaodan Liang, Liang Lin, and Xiaojun Chang. Blockwisely supervised neural architecture search with knowledge distillation. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2, 3, 4, 5, 6, 11, 12

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft COCO: Common objects in context. *European Conference on Computer Vision (ECCV)*, 2014. 5, 8

[20] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *International Conference on Learning Representations (ICLR)*, 2019. 1, 2

[21] Ilya Loshchilov and Frank Hutter. SGDR: Stochastic gradient descent with warm restarts. *International Conference on Learning Representations (ICLR)*, 2017. 11

[22] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. NSGANetV2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 6

[23] Zhichao Lu, Gautam Sreekumar, Erik Goodman, Wolfgang Banzhaf, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Neural architecture transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021. 1

[24] Jieru Mei, Yingwei Li, Xiaochen Lian, Xiaojie Jin, Linjie Yang, Alan Yuille, and Jianchao Yang. AtomNAS: Fine-grained end-to-end neural architecture search. *International Conference on Learning Representations (ICLR)*, 2020. 2

[25] Markus Nagel, Rana Ali Amjad, Marinus van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*. 2020. 4

[26] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. *International Conference on Computer Vision (ICCV)*, 2019. 15

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 12

[28] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. *International Conference on Machine Learning (ICML)*, 2018. 2

[29] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V. Le, and Alexey

Kurakin. Large-scale evolution of image classifiers. *International Conference on Machine Learning (ICML)*, page 2902–2911, 2017. 2

[30] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 11

[31] Dimitrios Stamoulis, Ruizhou Ding, Di Wang, Dimitrios Lymberopoulos, Bodhi Priyantha, Jie Liu, and Diana Marculescu. Single-path nas: Designing hardware-efficient convnets in less than 4 hours. In *arXiv preprint arXiv:1904.02877*, 2019. 1, 2

[32] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. MnasNet: Platform-aware neural architecture search for mobile. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 2, 5, 6, 7, 8, 12

[33] Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *International Conference on Machine Learning (ICML)*, 2019. 1, 3, 11

[34] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 8

[35] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, and Joseph E. Gonzalez. FBNetV2: Differentiable neural architecture search for spatial and channel dimensions. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[36] Ross Wightman. efficientdet-pytorch. https://github.com/rwightman/efficientdet-pytorch, 2020. 8

[37] Ross Wightman. pytorch-image-models. https://github.com/rwightman/pytorch-image-models, 2020. 5, 7, 8, 11, 12

[38] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. FBNet: Hardware-aware efficient convnet design via differentiable neural architecture search. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[39] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. *Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. 7, 15

[40] Yuhui Xu, Lingxi Xie, Xiaopeng Zhang, Xin Chen, Guo-Jun Qi, Qi Tian, and Hongkai Xiong. PC-DARTS: Partial channel connections for memory-efficient architecture search. *International Conference on Learning Representations (ICLR)*, 2020. 2

[41] Jiahui Yu, Pengchong Jin, Hanxiao Liu, Gabriel Bender, Pieter-Jan Kindermans, Mingxing Tan, Thomas Huang, Xiaodan Song, Ruoming Pang, and Quoc Le. BigNAS: Scaling up neural architecture search with big single-stage models. *European Conference on Computer Vision (ECCV)*, 2020. 1

[42] Barret Zoph and Quoc V. Le. Neural architecture search with reinforcement learning. *International Conference on Learning Representations (ICLR)*, 2017. 2

[43] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2