MUXConv: Information Multiplexing in Convolutional Neural Networks

Zhichao Lu Kalyanmoy Deb Vishnu Naresh Boddeti
Michigan State University
{luzhicha, kdeb, vishnu}@msu.edu

Abstract

Convolutional neural networks have witnessed remarkable improvements in computational efficiency in recent years. A key driving force has been the idea of trading-off model expressivity and efficiency through a combination of 1×1 and depth-wise separable convolutions in lieu of a standard convolutional layer. The price of the efficiency, however, is the sub-optimal flow of information across space and channels in the network. To overcome this limitation, we present MUXConv, a layer that is designed to increase the flow of information by progressively multiplexing channel and spatial information in the network, while mitigating computational complexity. Furthermore, to demonstrate the effectiveness of MUXConv, we integrate it within an efficient multi-objective evolutionary algorithm to search for the optimal model hyper-parameters while simultaneously optimizing accuracy, compactness, and computational efficiency. On ImageNet, the resulting models, dubbed MUXNets, match the performance (75.3% top-1 accuracy) and multiply-add operations (218M) of MobileNetV3 while being 1.6× more compact, and outperform other mobile models in all the three criteria. MUXNet also performs well under transfer learning and when adapted to object detection. On the ChestX-Ray 14 benchmark, its accuracy is comparable to the state-of-the-art while being 3.3× more compact and 14× more efficient. Similarly, detection on PASCAL VOC 2007 is 1.2% more accurate, 28% faster and 6% more compact compared to MobileNetV2. The code is available from https://github.com/human-analysis/MUXConv.

1. Introduction

In the span of the last decade, convolutional neural networks (CNNs) have undergone a dramatic transformation in terms of predictive performance, compactness and computational efficiency. The development largely happened in two phases. Starting from AlexNet [20], the focus of the first wave of models was on improving the predictive accuracy of CNNs including VGG [35], GoogleNet [37], ResNet [11], ResNeXt [43], DenseNet [16] etc. These models progressively increased the contribution of 3×3 convolutions, both in model size as well as multiply-add operations (MAdds). The focus of the second wave of models was on improving their computational efficiency while trading-off accuracy to a small extent. Models in this category include ShuffleNet [26], MobileNetV2 [32], MnasNet [38] and MobileNetV3 [12]. Such solutions sought to improve computational efficiency by progressively replacing the parameter and compute intensive standard convolutions by a combination of 1×1 convolutions and depth-wise separable 3×3 convolutions. Figure 2 depicts the trend in the relative contributions of different layers in terms of parameters and MAdds.

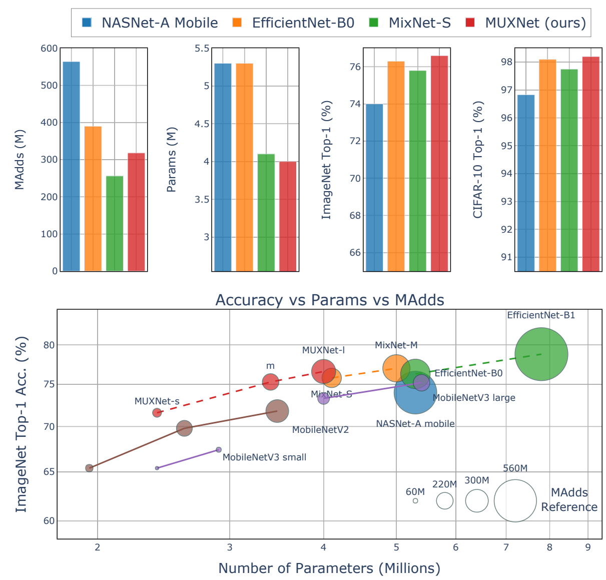


Figure 1: Accuracy vs. Compactness vs. Efficiency: Existing networks outperform each other in at most two criteria. MUXNet models are, however, dominant in all three objectives under mobile settings.

Depth-wise separable convolutions [34, 4] offer significant computational benefits, both from the perspective of number of parameters as well as computational complexity. A salient feature of these layers is the lack of interac-

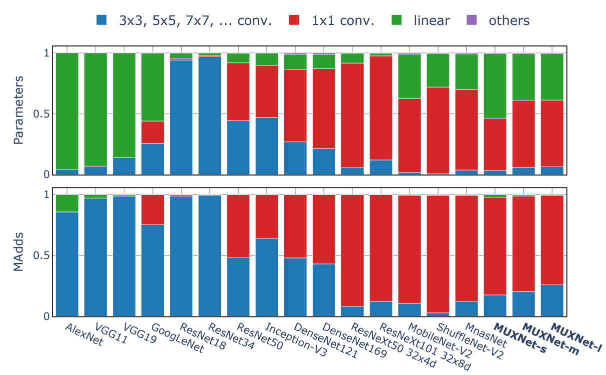


Figure 2: Relative contribution of different layers in CNN designs in terms of parameters (top) and MAdds (bottom). Initial models largely relied on standard convolutional layers. More recent networks, on the other hand, largely rely on 1×1 convolutions and linear layers. In contrast, MUXNets reverse this trend to an extent.

tions between information in the channels. This limitation is overcome through 1×1 convolution, a layer which allows for interactions and information flow across the channels. The combination of depth-wise separable and 1×1 convolution fully decouples the task of spatial and channel information flow, respectively, into two independent and efficient layers. On the other hand, a standard convolutional layer couples the spatial and channel information flow into a single, yet, computationally inefficient layer. Therefore, the former replaced the latter as the workhorse of CNN designs.

In this paper, we seek an alternative approach to trade-off the expressivity and efficiency of convolutional layers. We introduce MUXConv, a layer that leverages the efficiency of depth-wise or group-wise convolutional layers along with a mechanism to enhance the flow of information in the network. MUXConv achieves this through two components, spatial multiplexing and channel multiplexing. Spatial multiplexing extracts feature information at multiple scales via spatial shuffling, processes such information through depth-wise or group-wise convolutions and then unshuffles them back together. Channel multiplexing is inspired by ShuffleNet [26] and is designed to address the limitation of depth-wise/group convolutions, namely the lack of information flow across channels/groups of channels, by shuffling the channels. The shuffling procedure and the operations we perform on the shuffled channels are motivated by computational efficiency and differ significantly from ShuffleNet. Collectively, these two components increase the flow of information, both spatially and across channels, while mitigating the computational burden of the layer.

To further realize the full potential of MUXConv in trading-off accuracy and computational efficiency, we propose a population based evolutionary algorithm to efficiently search for the hyperparameters of each MUXConv layer in the network. The search simultaneously optimizes three objectives, namely, prediction accuracy, model compactness and model efficiency in terms of MAdds. To improve the efficiency of the search process we decompose the multi-objective optimization problem into a collection of single-objective optimization sub-problems, that are in turn optimized simultaneously and cooperatively. We refer to the resulting family of CNNs as MUXNets.

Contributions: We first develop a new layer, called MUXConv, that multiplexes information flow spatially and across channels while improving the computational efficiency of equivalent combination of depth-wise separable and 1×1 convolutions. Then, we develop the first multi-objective neural architecture search (NAS) algorithm to simultaneously optimize compactness, efficiency, and accuracy of MUXNets designed with MUXConv as the basic building block. We present thorough experimental evaluation demonstrating the efficacy and value of each component of MUXNet across multiple tasks including image classification (ImageNet), object detection (PASCAL VOC 2007) and transfer learning (CIFAR-10, CIFAR-100, ChestX-Ray14). Our results indicate that, unlike the conventional wisdom in all existing solutions, it is feasible to design CNNs that do not sacrifice compactness for efficiency or vice versa in the quest for better predictive performance.

2. Related-work

Many CNN architectures have been developed by optimizing different objectives, such as, model compactness, computational efficiency, or predictive performance. Below, we categorize the solutions into a few major themes.

Multi-Scale and Shuffling: The notion of multi-scale processing in CNNs has been utilized in different forms and in a variety of contexts. These include explicit processing of multi-resolution feature maps for object detection [2, 21] and image classification [14] and computational blocks with built-in multi-scale processing [3, 9]. The focus of these methods is predictive performance and hence towards large scale models. In contrast, multi-scale processing in MUXConv is motivated by enhancing information flow in small scale models deployed in resource constrained environments. Notably, MUXConv scales the feature maps through a pixel shuffling operation that is similar to subpixel convolution in [33]. The channel shuffling component of MUXConv is motivated by [47, 26].

Mobile Architectures: A number of CNN architectures have been developed for mobile settings. These include SqueezeNet [18], MobileNet [13], MobileNetV2 [32], MobileNetV3 [12], ShuffleNet [47], ShuffleNetV2 [26] and CondenseNet [15]. The focus of this body of work has largely been to optimize two objectives, either accuracy and compactness or accuracy and efficiency, thereby resulting in models that are either efficient or compact but not both. In contrast, MUXNets are designed to simultaneously optimize all three objectives, compactness, efficiency and accuracy, and therefore leads to models that are both compact and efficient at the same time.

Neural Architecture Search: Automated approaches to search for good neural architectures have proven to be very effective in finding computational blocks that not only exhibit high predictive performance but also generalize and transfer to other tasks. Majority of the approaches including, NasNet [48], PNAS [22], DARTS [23], AmoebaNet [30] and MixNet [40], are optimized against a single objective, namely predictive performance. A couple of recent approaches, LEMONADE [7], NSGANet [25], simultaneously optimize the networks against multiple objectives, including parameters, MAdds, latency, and accuracy. However, only results on small-scale datasets like CIFAR-10 are demonstrated in both approaches. Concurrently, a number of CNN architectures, such as ProxylessNAS [1], MnasNet [38], ChamNet [5] and FBNet [5], have been designed to target specific computing platforms such as mobile, CPU, and GPU. In contrast to the aforementioned NAS approaches, we adopt a hybrid search strategy where the basic computational block, MUXConv, is hand-designed while the hyper-parameters of each MUXConv layer in the network are searched through a population based evolutionary algorithm directly on a large scale dataset.

3. Multiplexed Convolutions

The multiplexed convolution layer, called MUXConv, is a combination of two components: (1) spatial multiplexing which enhances the expressivity and predictive performance of the network, and (2) channel multiplexing which aids in reducing the computational complexity of the model.

3.1. Spatial Multiplexing

The expressivity of a standard convolutional layer stems from the flow of information spatially and across the channels. Spatial multiplexing is designed to mimic this property while mitigating its computational complexity. The key idea is to map spatial information at multiple scales into channels and vice versa. Specifically, given a feature map $\boldsymbol{x} \in \mathbb{R}^{C\times H\times W}$, where C is the number of channels, H is the height and W is the width of the feature map, the channels are grouped into three groups of (C_1, C_2, C_3) channels such that $C = C_1 + C_2 + C_3$. The first and third group of channels are subjected to a subpixel and superpixel multiplexing operation, respectively. The multiplexed channels are then processed through a group-wise convolution operation defined over each of the three groups. The output feature maps from the group convolutions are mapped back to the same dimensions as the input feature maps by reversing

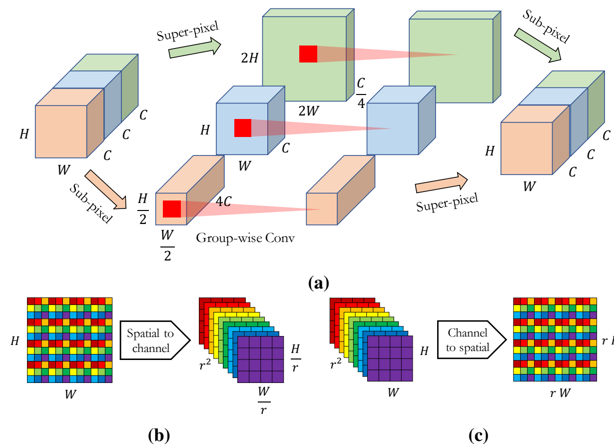


Figure 3: (a) Overview of spatial multiplexing operation. (b) Subpixel operation multiplexes spatial information into channels. (c) Superpixel operation multiplexes channels into spatial information.

the respective subpixel and superpixel operations. An illustration of this process is shown in Fig. 3a. Collectively, the subpixel and superpixel operations allow multi-scale spatial information to flow across channels. We note that the standard idea of multi-scale processing in existing approaches, multi-scale feature representations or kernels with larger receptive fields, is typically across different layers. In contrast, MUXConv seeks to exploit multi-scale information within a layer through pixel manipulation. As we show in Section 6, this operation significantly improves network accuracy especially as they get more compact.

We parameterize the subpixel multiplexing operation (see Fig. 3b) by r and define a window and stride of size $r{\times}r$. The features in the windows are mapped to r^2 channels, with each window corresponding to a unique feature location in the channels. On the whole, the subpixel operation maps the first group of channel features of size $C_1 \times H \times W$ to features of size $r^2 C_1 \times \frac{H}{r} \times \frac{W}{r}$. Therefore, the subpixel operation enables down-scaled spatial information to be multiplexed with channel information and processed jointly by a standard convolution over the group. The combination of the two operations effectively increases the receptive field of the convolution by a factor of r.

We define the superpixel multiplexing operation (see Fig. 3c) as an inverse of subpixel multiplexing. It is parameterized by r^2 which corresponds to the number of channels that will be multiplexed spatially into a single channel. The feature values at a particular location from the r^2 channels are mapped to a unique window in the output feature map. On the whole, the superpixel operation maps the third group of channels features of size $C_3 \times H \times W$ to features of size $\frac{C_3}{r^2} \times rH \times rW$. Therefore, the superpixel operation enables channel information to be multiplexed with up-scaled spatial information and processed jointly by a standard convolution over the group. The combination of the two oper-

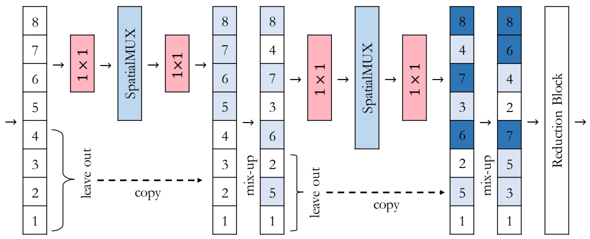


Figure 4: Illustration of two channel multiplexing layers. In each layer, half the channels are propagated as is while the other half are processed through the spatial multiplexing operation. The channels from the two groups are then interleaved as denoted by the indices. Color intensity denotes number of times that channel is processed.

ations effectively decreases the receptive field of the convolution by a factor of r. Our superpixel operation bears similarity to the concept of *tiled convolution* [27], a particular realization of locally connected layers. This idea has also been particularly effective for image super-resolution [33] in the form of "subpixel" convolution.

3.2. Channel Multiplexing

While the spatial multiplexing operation described above is effective, it still suffers from some limitations. Firstly, the group convolutions in spatial multiplexing are more computationally expensive than depth-wise separable convolutions that they replace. Secondly, the decoupled nature of the group convolutions does not allow for flow of information across the groups. The channel multiplexing operation is designed to mitigate these drawbacks by reducing the computational burden of spatial multiplexing and further enhancing the flow of information across the feature map channels. This is achieved in two stages, selective processing and channel shuffling. A illustration of the whole operation is shown in Fig. 4. Overall, the channel multiplexing operation is similar in spirit to ShuffleNet [47] and ShuffleNetV2 [26] but with notable variations; (1) ShuffleNet uses shuffling to share channel information that are processed in different groups, while we use shuffling to blend the raw and processed channel information., (2) While ShuffleNetV2 always splits the input channels in half, we treat it as a hyperparameter that is searched for each layer, and (3) Shuffled channels are processed through an inverted residual bottleneck block in ShuffleNetV2 as opposed to spatial multiplexing in our case.

Selective Processing: We process only a part of the input channels by the spatial multiplexing block. Specifically, the C channels in the input feature maps are split into two groups with C_1 and C_2 channels, such that $C = C_1 + C_2$. The first group of channels are propagated as is while the second group are processed through spatial multiplexing. This scheme immediately increases the compactness and efficiency by a factor of $\left(\frac{C}{C_2}\right)^2$, which can compensate for the computational burden of grouped as opposed to depth-wise separable convolutions.

Channel Shuffling: After the selective processing operation, we shuffle the channels of the output feature map in a fixed pattern. Alternative channels selected from the unprocessed and processed channels are interleaved.

4. Tri-Objective Hyperparameter Search

Designing a CNN typically involves many hyperparameters that critically impact the performance of the models. In order to realize the full potential of MUXNet we seek to search for the optimal hyperparameters in each layer of the network. Since the primary design motive of MUXConv is to increase model expressivity while mitigating computational complexity, we propose a multi-objective hyperparameter search algorithm to simultaneously optimize for accuracy, compactness and efficiency. This can be stated as,

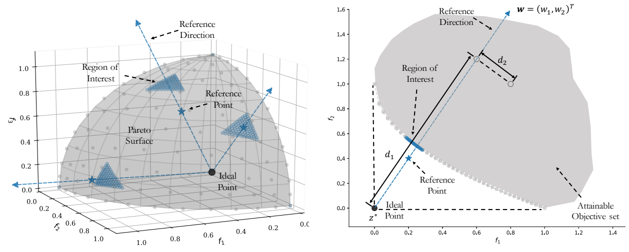
$$\begin{aligned} \text{minimize} \quad & \mathbf{F}(\boldsymbol{x}) = \big(f_1(\boldsymbol{x}), \cdots, f_m(\boldsymbol{x})\big)^T, \\ \text{subject to} \quad & \boldsymbol{x} \in \boldsymbol{\Omega}, \end{aligned} \tag{1}$$

where in our context $\boldsymbol{\Omega} = \Pi_{i=1}^{n}[a_i, b_i] \subseteq \mathbb{R}^n$ is the hyperparameter decision space, where a_i, b_i are the lower and upper bounds, $\boldsymbol{x} = (x_1, \ldots, x_n)^T \in \boldsymbol{\Omega}$ is a candidate hyperparameter setting, $\mathbf{F} : \boldsymbol{\Omega} \rightarrow \mathbb{R}^m$ constitutes m competing objectives, i.e. predictive error, model size, model inefficiency, etc., and $\mathbb{R}^m$ is the objective space.

As the number of objectives increases, the number of solutions needed to approximate the entire Pareto surface grows exponentially [6], rendering a global search impractical in most cases. To overcome this challenge we propose a reference guided hyperparameter search. Instead of spanning the entire search space, we focus the hyperparameter search to a neighborhood around few desired user-defined preferences. An illustration of this concept is shown in Fig. 5a. For instance, in our context, this could correspond to different desired accuracy targets and hardware specifications. This idea enables us to decompose the tri-objective problem into multiple single objective subproblems. We adopt the penalty-based boundary intersection (PBI) method [46] to scalarize multiple objectives into a single objective,

$$\begin{aligned} \text{minimize} \quad & g^{pbi}(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{z}^*) = d_1 + \theta d_2 \\ \text{subject to} \quad & \boldsymbol{x} \in \boldsymbol{\Omega}, \end{aligned} \tag{2}$$

where $d_2 = \left\| \mathbf{F}(\boldsymbol{x}) - \left(\boldsymbol{z}^* + d_1 \frac{\boldsymbol{w}}{||\boldsymbol{w}||}\right) \right\|$, $d_1 = \frac{||(\mathbf{F}(\boldsymbol{x}) - \boldsymbol{z}^*)^T \boldsymbol{w}||}{||\boldsymbol{w}||}$, $\boldsymbol{z}^* = (z_1^*, \ldots, z_m^*)^T$ is the ideal objective vector with $z_i^* < \min_{\boldsymbol{x} \in \boldsymbol{\Omega}} f_i(\boldsymbol{x})\ i \in \{1, \ldots, m\}$. $\theta \geq 0$ is a trade-off hyperparameter that is set to 5 and $\boldsymbol{w}$ is the reference direction obtained by connecting the ideal solution to the desired reference target.



(a) **(b)**

Figure 5: Tri-Objective Search: (a) We leverage user-defined preferences to decompose the tri-objective problem into multiple single-objective sub-problems. By focusing on sub-regions as opposed to the entire Pareto surface, our approach is more efficient. (b) The reference direction is formed by joining the ideal point and user supplied reference targets. The PBI method is used to scalarize the objectives based on the projected distance d_2 to the reference target $\boldsymbol{w}$, and the distance d_1 to the ideal point.

Conceptually, the PBI method constructs a composite measure of the convergence (d_1) of the solution to the given reference targets and diversity (d_2) of the solutions itself. See Fig.5b for an illustration. In our context, d_1 (distance between current projected solution and ideal solution) seeks to push the solution to the boundary of attainable objective space and d_2 measures how close the solution is to the user's preference. Finally, we adopt a multi-objective evolutionary algorithm based on decomposition (MOEA/D [46]), to simultaneously solve the decomposed sub-problems while optimizing the scalarized objective.

5. Experiments

We evaluate the efficacy of MUXNets on three tasks; image classification, object detection, and transfer learning.

5.1. Hyperparameter Search Details

Search Space: To compensate for the extra hyperparameters introduced by spatial and channel multiplexing, we constrain the commonly adopted layer-wise search space [1, 38, 12] to a stage-wise search space, where layers within the same stage share the same hyperparameters. MUXNets consist of four stages, where each stage begins with a reduction block and is followed by a series of normal blocks. In each stage, we search for kernel size, expansion ratio, repetitions of normal blocks, leave-out ratio for channel multiplexing and the spatial multiplexing settings (see supplementary for details). To further reduce the search space, we always adopt squeeze-and-excitation [18] and use swish [29] non-linearity for activation at each stage except the first stage, where a ReLU is used.

Search: Following previous work [1, 38], we conduct the search directly on ImageNet and estimate model accuracy on a subset consisting of 50K randomly sampled images from the training set. As a common practice, during search, the number of training epochs are reduced to 5. We select four reference points with preferences on model size ranging from 1.5M to 5M, MAdds ranging from 60M to 300M, and predictive accuracy fixed at 1. The compactness and efficiency objectives are normalized between [0, 1] before aggregation. Search is initialized with a global population size of 40 and evolved for 100 iterations, which takes about 11 days on sixteen 2080Ti GPUs. At the end of evolution, we pick the top 5 (based on PBI aggregated function values) models from each of the four subproblems, and retrain them thoroughly from scratch on ImageNet. The four resulting models are named as MUXNet-xs/s/m/l. Architectural details can be found in the supplementary material.

5.2. ImageNet Classification

For training on ImageNet, we follow the procedure outlined in [38]. Specifically, we adopt Inception preprocessing with image size 224×224 [36], batch size of 256, RMSProp optimizer with decay 0.9, momentum 0.9, and weight decay 1e-5. A Dropout layer of rate 0.2 is added before the last linear layer. Learning rate is linearly increased to 0.016 in the initial 5 epochs [10], it then decays every 3 epochs at a rate of 0.03. We further complement the training with exponential moving average with decay rate of 0.9998.

Table 1 shows the performance of baselines and MUXNets on ImageNet 2012 benchmark [31]. We compare them in terms of accuracy on validation set, model compactness (parameter size), model efficiency (MAdds) and inference latency on CPU and GPU. Overall, MUXNets consistently either match or outperform other models across different accuracy levels. In particular, MUXNet-m achieves 75.3% accuracy with 3.4M parameters and 218M MAdds, which is **1.4**× more efficient and **1.6**× more compact when compared to MnasNet-A1 [38] and MobileNetV3 [12], respectively. Figures 1 and 6 visualize the trade-off obtained by MUXNet and previous models. In terms of accuracy and compactness, MUXNet clearly dominates all previous models including MnasNet [38], FBNet [42], MobileNetV3 [12], and MixNet [40]. In terms of accuracy and efficiency, MUXNets are on par with current state-of-the-art models, i.e. MobileNetV3 and MixNet.

In terms of latency, the performance of MUXNet models is mixed since they, (i) use non-standard primitives that do not have readily available efficient low-level implementations, and (ii) are not explicitly optimized for latency. Compared to methods that use optimized convolutional primitives but do not directly optimize for latency (EfficientNet/MixNet), MUXNet's latency is competitive despite using unoptimized spatial and channel multiplexing primitives. MUXNet's limitations due to unoptimized implementation can be offset, to an extent, by its inherent FLOPs and parameter efficiency. MUXNet is not as competitive as methods that directly use CPU or GPU latency on Pixel phones as a search objective (MobileNetV3, MnasNet).

Table 1: ImageNet Classification [31]: MUXNet comparison with manual and automated design of efficient convolutional neural networks. Models are grouped into sections for better visualization. Our results are underlined and the best result in each section is in bold. CPU latency (batchsize=1) is measured on Intel i7-8700K and GPU latency (batchsize=64) is measured on 1080Ti. ‡ indicates the objective (in addition to predictive performance) that the method explicitly optimizes through NAS.

Model	Type	#MAdds	Ratio	#Params	Ratio	CPU(ms)	GPU(ms)	Top-1 (%)	Top-5 (%)
MUXNet-xs (ours)	auto	**66M**‡	1.0x	**1.8M**‡	1.0x	6.8	18	66.7	86.8
MobileNetV2_0.5 [32]	manual	97M	1.5x	2.0M	1.1x	**6.2**	17	65.4	86.4
MobileNetV3 small [12]	combined	**66M**	1.0x	2.9M	1.6x	**6.2**‡	**14**	**67.4**	-
MUXNet-s (ours)	auto	**117M**‡	1.0x	**2.4M**‡	1.0x	9.5	25	**71.6**	90.3
MobileNetV1 [13]	manual	575M	4.9x	4.2M	1.8x	7.3	20	70.6	89.5
ShuffleNetV2 [26]	manual	146M	1.3x	-	-	**6.8**	**11**‡	69.4	-
ChamNet-C [5]	auto	212M	1.8x	3.4M	1.4x	-	-	**71.6**	-
MUXNet-m (ours)	auto	**218M**‡	1.0x	**3.4M**‡	1.0x	14.7	42	**75.3**	92.5
MobileNetV2 [32]	manual	300M	1.4x	3.4M	1.0x	**8.3**‡	23	72.0	91.0
ShuffleNetV2 2× [26]	manual	591M	2.7x	7.4M	2.2x	11.0	**22**‡	74.9	-
MnasNet-A1 [38]	auto	312M	1.4x	3.9M	1.1x	9.3‡	32	75.2	92.5
MobileNetV3 large [12]	combined	219M	1.0x	5.4M	1.6x	10.0‡	33	75.2	-
MUXNet-l (ours)	auto	**318M**‡	1.0x	**4.0M**‡	1.0x	19.2	74	76.6	93.2
MnasNet-A2 [38]	auto	340M	1.1x	4.8M	1.2x	-	-	75.6	92.7
FBNet-C [42]	auto	375M	1.2x	5.5M	1.4x	**9.1**‡	**31**	74.9	-
EfficientNet-B0 [39]	auto	390M‡	1.2x	5.3M	1.3x	14.4	46	76.3	93.2
MixNet-M [40]	auto	360M‡	1.1x	5.0M	1.2x	24.3	79	**77.0**	93.3

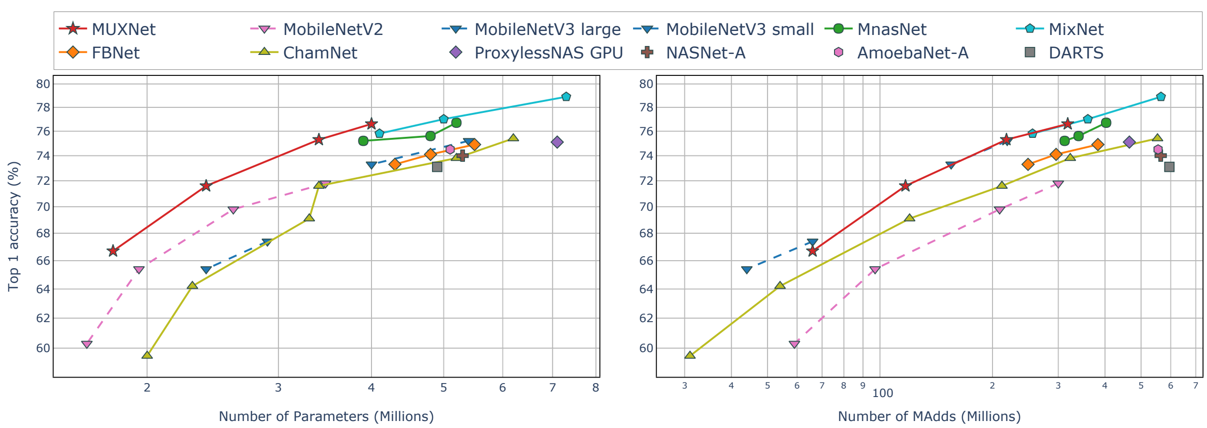


Figure 6: The trade-off between model complexity and top-1 accuracy on ImageNet. This allows us to compare models designed for different computation requirements in number of parameters or number of multi-adds. All our models use input resolution of 224×224. We use dash line to denote models from channel width multipliers or with different input resolutions.

5.3. Object Detection

Table 2: PASCAL VOC2007 [8] Detection

Network	#MAdds	#Params	mAP (%)
VGG16 + SSD [24]	35B	26.3M	74.3
MobileNet + SSD [17]	1.6B	9.5M	67.6
MobileNetV2 + SSDLite [32]	0.7B	3.4M	67.4
MobileNetV2 + SSD [32]	1.4B	8.9M	73.2
MUXNet-m + SSDLite (ours)	**0.5B**	**3.2M**	**68.6**
MUXNet-l + SSD (ours)	**1.4B**	**9.9M**	**73.8**

We evaluate and compare the generalization ability of MUXNet and other peer models on the PASCAL VOC detection benchmark [8]. Our experiments use both the Single Shot Detector (SSD) [24] and the Single Shot Detector Lite (SSDLite) [32] as the detection frameworks, with MUXNet as the feature extraction backbone. We follow the procedure in [32] to setup the additional prediction layers, i.e. location of detection heads in the backbone, size of corresponding boxes, etc. The combined *trainval* sets of PASCAL VOC 2007 and 2012 are used for training. Other details include, SGD optimizer with momentum 0.9 and weight decay 5e-4, batch size of 32, input image resized to 300×300 and learning rate of 0.01 with cosine annealing to 0.0 in 200 epochs. Table 2 reports the mean Average Precision (mAP) on the PASCAL VOC 2007 test set. When paired with the

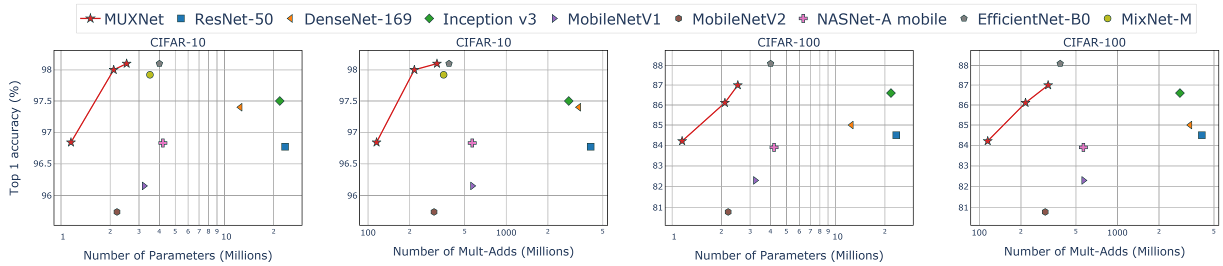


Figure 7: Transfer Learning on CIFAR: Trade-off between Top-1 accuracy and #Params / #MAdds.

same detector framework SSDLite, our MUXNet-m model achieves 1.2% higher mAP than MobileNetV2 [32] while being 6% more compact and $1.4\times$ more efficient.

5.4. Transfer Learning

To further explore the efficacy of MUXNet we evaluate it under the transfer learning setup in [19] on three different datasets; CIFAR-10, CIFAR-100 and ChestX-Ray14 [41].

5.4.1 CIFAR-10 and CIFAR-100

Both CIFAR-10 and -100 datasets have 50,000 and 10,000 images for training and testing, respectively. CIFAR-100 extends CIFAR-10 by adding 90 more classes resulting in $10\times$ fewer training examples per class. For training on both datasets, the models are initialized with weights pre-trained on ImageNet. The model is then fine-tuned using SGD with momemtum 0.9, weight decay 4e-5 and gradients clipped to a magnitude of 5. Learning rate is set to 0.01 with cosine annealing to 0.0 in 150 epochs. For data augmentation, images are up-sampled via bicubic interpolation to 224×224 and horizontally fliped at random. Table 3 and Figure 7 reports the accuracy, compactness and efficiency of MUXNet and other baselines. Overall, MUXNet significantly outperforms previous methods on both CIFAR-10 and -100 datasets, indicating that our models also transfer well to other similar tasks. In particular, MUXNet-m achieves 1% higher accuracy than NASNet-A mobile with $\mathbf{3}\times$ fewer parameters while being $\mathbf{2}\times$ more efficient in MAdds.

5.4.2 ChestX-Ray14

The ChestX-Ray14 benchmark was recently introduced in [41]. The dataset consists of 112,120 high resolution frontal-view chest X-ray images from 30,805 patients. Each image is labeled with one or multiple common thorax diseases, or "Normal", otherwise. Due to the multi-label nature of the dataset, we use a multitask learning setup where each disease is treated as an individual binary classification problem. We define a 14-dimensional label vector of binary values indicating the presence of one or more diseases, and optimize a regression loss as opposed to cross-entropy in single-label cases. The training procedure is similar to the CIFAR experiments for transfering pre-trained models. Table 4 compares the performance of MUXNet-m with previous approaches, including CheXNet [28] which represents the state-of-the-art on this dataset. Evidently, MUXNet-m's performance in terms of area under the receiver operating characteristic (AUROC) curve on the test set is comparable (84.1% vs 84.4%) to CheXNet while being $\mathbf{3}\times$ more compact and $\mathbf{14}\times$ more efficient.

Table 3: Transfer Learning: Top-1 accuracy on CIFAR-10 (C-10) and CIFAR-100 (C-100). ResNet, DenseNet, MobileNetV2, and NASNet-A results are from [19].

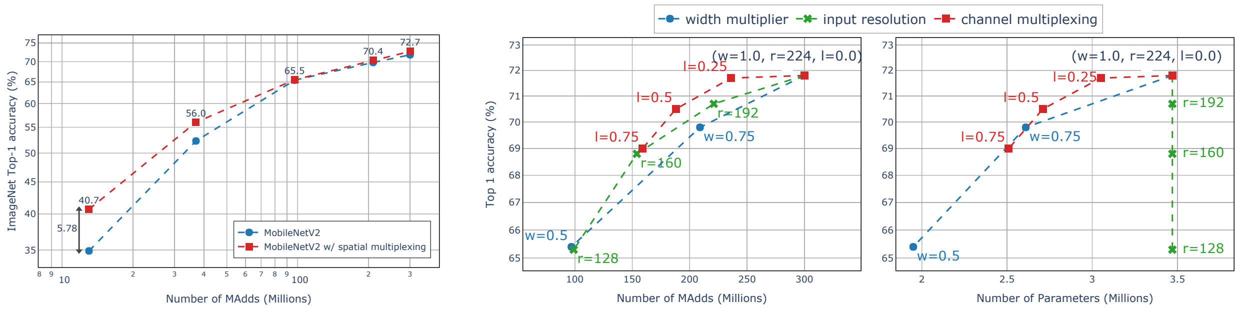
Model	#MAdds	#Params	C-10 (%)	C-100 (%)
ResNet-50 [11]	4.1B	23.5M	96.77	84.50
DenseNet-169 [16]	3.4B	12.5M	97.40	85.00
MobileNetV2 [32]	0.3B	2.2M	95.74	80.80
NASNet-A mobile [48]	0.6B	4.2M	96.83	83.90
EfficientNet-B0 [39]	0.4B	4.0M	98.10	88.10
MixNet-M [40]	0.4B	3.5M	97.92	-
MUXNet-m (ours)	**0.2B**	**2.1M**	**98.00**	**86.11**

Table 4: Transfer Learning on ChestX-Ray14 [41]

Method	#MAdds	#Params	Test AUROC (%)
Wang et al. (2017) [41]	-	-	73.8
Yao et al. (2017) [44]	-	-	79.8
CheXNet (2017) [28]	2.8B	7.0M	84.4
MUXNet-m (ours)	**0.2B**	**2.1M**	**84.1**

6. Ablation Study

Spatial Multiplexing: We incorporate the spatial multiplexing operation within the 3×3 depth-wise separable convolution layers of MobileNetV2. As we do in our main experiments, we do not apply spatial multiplexing to the reduction blocks. We manually fix the multiplexing hyperparameters to $C_1 = C_3 = \frac{C}{4}, C_2 = \frac{C}{2}$ i.e., 1/4 channels are processed by subpixeling, 1/4 of the channels are processed by superpixeling, and the remaining channels are processed without modification. Figure 8a shows the effect of spatial multiplexing on MobileNetV2 [32] at different width multipliers. Spatial multiplexing consistently improves accuracy over the original depth-wise separable convolution at fixed



(a) Spatial Multiplexing

(b) Channel Multiplexing

Figure 8: Multiplexed Convolution Ablation Study: (a) Results correspond to width multiplier of 0.1, 0.25, 0.5, 0.75, and 1.0. (b) w, r and l are width multiplier, input resolution and leave-out ratio, respectively. When $l = 0.25$, 75% of the input information is processed at each normal block.

spatial resolution. In particular, spatial multiplexing boosts accuracy by **5.8**% in low MAdds regime. The results suggest that per MAdd, spatial multiplexing (groups+full conv) has better information flow than dep-sep+1 $\times$ 1 conv. This is more apparent in small models which have less channels, so 1 $\times$ 1 conv cannot effectively mix channel information.

Channel Multiplexing: To make models more efficient, methods such as scaling down the number of channels by a factor (named width multiplier), or scaling down the input resolution have been proposed. Here we investigate the impact of channel multiplexing as an alternative to reduce model complexity. To be consistent with the main experiments we only apply channel multiplexing to the normal blocks. In MobileNetV2 [32] we gradually increase the number of input channels that are left unprocessed in each normal block. We use l to denote the leave-out ratio, where a high value corresponds to less channels being processed and hence more efficiency. The resulting trade-off with accuracy is shown in Figure 8b. Evidently, reducing the resolutions of input images provides a better trade-off between accuracy and MAdds than reducing the channels. However, reducing the input resolution provides no benefit to model size. On the other hand, channel multiplexing offers competitive trade-off in both cases; MAdds and model size. In particular, leaving out 25% of the input channels at every normal block appears to affect the predictive accuracy minimally, while simultaneously saving **13**% in parameters and **20**% in multiply-adds.

Search Efficiency: To thoroughly and efficiently evaluate the effectiveness of the PBI decomposition technique and the search efficiency of our proposed NAS algorithm, we adopt the NASBench101 [45] benchmark. It contains more than 400K unique models pre-trained on CIFAR-10, whose Pareto-optimal solutions and predictive performance are readily available without expensive training. In this case, we aim to minimize the number of parameters, the training time and maximize the accuracy. We also adopt the regularized evolution [30] approach as a baseline for comparison. Figure 9 shows the search effectiveness for three

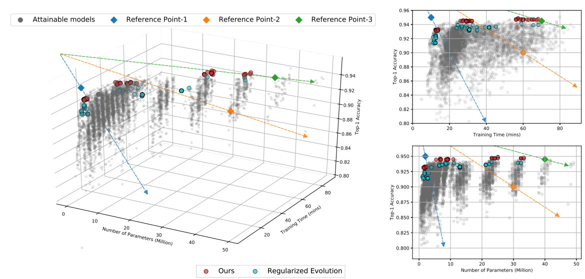


Figure 9: Performance comparison between our approach and regularized evolution (RE) [30] on NASBench101 [45]. Both methods are subject to the same search budget of 1,000 maximum models sampled. We distribute the search budget across three executions of RE for each one of the three reference points. Our approach simultaneously targets all three reference points in one run using all available budget.

reference points under a fixed computational budget. The PBI scalarization is effective in directing the search towards pre-defined target regions as the obtained solutions from both methods are centered around the three provided target points. In addition, we observe that by collectively solving the sub-problems, we achieve better results under the same search budget as opposed to solving the sub-problem one at a time, as in case of regularized evolution.

7. Conclusion

This paper introduced MUXConv, an efficient alternative to a standard convolutional layer that is designed to progressively multiplex channel and spatial information in the network. Furthermore, we coupled it with an efficient multi-objective evolutionary algorithm based hyperparameter search to trade-off predictive accuracy, model compactness and computational efficiency. Experimental results on image classification, object detection and transfer learning suggest that MUXNets are able to match the predictive accuracy and efficiency of current state-of-the-art models while be more compact.

Acknowledgements: We gratefully acknowledge Dr. Erik Goodman and Dr. Wolfgang Banzhaf for partially supporting the computational requirements of this work.

References

[1] Han Cai, Ligeng Zhu, and Song Han. ProxylessNAS: Direct neural architecture search on target task and hardware. In *International Conference on Learning Representations (ICLR)*, 2019. 3, 5

[2] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *European Conference on Computer Vision (ECCV)*, 2016. 2

[3] Chun-Fu Chen, Quanfu Fan, Neil Mallinar, Tom Sercu, and Rogerio Feris. Big-little net: An efficient multi-scale feature representation for visual and speech recognition. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[4] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[5] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 3, 6

[6] K. Deb. *Multi-objective optimization using evolutionary algorithms*. Chichester: Wiley, 2001. 4

[7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[8] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan 2015. 6

[9] Shang-Hua Gao, Ming-Ming Cheng, Kai Zhao, Xin-Yu Zhang, Ming-Hsuan Yang, and Philip Torr. Res2net: A new multi-scale backbone architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 2

[10] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 5

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 1, 7

[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. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2, 5, 6

[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. *arXiv preprint arXiv:1704.04861*, 2017. 2, 6

[14] G Huang, D Che, T Li, F Wu, L van der Maaten, and K Weinberger. Multi-scale dense networks for resource efficient image classification. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[15] Gao Huang, Shichen Liu, Laurens Van der Maaten, and Kilian Q Weinberger. Condensenet: An efficient densenet using learned group convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[16] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 7

[17] Jonathan Huang, Vivek Rathod, Chen Sun, Menglong Zhu, Anoop Korattikara, Alireza Fathi, Ian Fischer, Zbigniew Wojna, Yang Song, Sergio Guadarrama, et al. Speed/accuracy trade-offs for modern convolutional object detectors. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 6

[18] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size. In *International Conference on Learning Representations (ICLR)*, 2016. 2, 5

[19] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 7

[20] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2012. 1

[21] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 2

[22] Chenxi Liu, Barret Zoph, Maxim Neumann, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[24] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision (ECCV)*, 2016. 6

[25] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: Neural architecture search using multi-objective genetic algorithm. In *Genetic and Evolutionary Computation Conference (GECCO)*, 2019. 3

[26] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *European Conference on Computer Vision (ECCV)*, 2018. 1, 2, 4, 6

[27] Jiquan Ngiam, Zhenghao Chen, Daniel Chia, Pang W Koh, Quoc V Le, and Andrew Y Ng. Tiled convolutional neural

networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2010. 4

[28] Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, et al. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *arXiv preprint arXiv:1711.05225*, 2017. 7

[29] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017. 5

[30] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In *AAAI Conference on Artificial Intelligence*, 2019. 3, 8

[31] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5, 6

[32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 6, 7, 8

[33] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2, 4

[34] Laurent Sifre and Stéphane Mallat. Rigid-motion scattering for image classification. *Ph. D. dissertation*, 2014. 1

[35] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-scale Image Recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 1

[36] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI Conference on Artificial Intelligence*, 2017. 5

[37] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 1

[38] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3, 5, 6

[39] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning (ICML)*, 2019. 6, 7

[40] Mingxing Tan and Quoc V. Le. Mixconv: Mixed depthwise convolutional kernels. In *British Machine Vision Conference (BMVC)*, 2019. 3, 5, 6, 7

[41] Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 7

[42] 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. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 6

[43] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1

[44] Li Yao, Eric Poblenz, Dmitry Dagunts, Ben Covington, Devon Bernard, and Kevin Lyman. Learning to diagnose from scratch by exploiting dependencies among labels. *arXiv preprint arXiv:1710.10501*, 2017. 7

[45] Chris Ying, Aaron Klein, Esteban Real, Eric Christiansen, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning (ICML)*, 2019. 8

[46] Qingfu Zhang and Hui Li. Moea/d: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007. 4, 5

[47] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2, 4

[48] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 3, 7