MixConv Applied to Classifying Diseased Alzheimers Brains

Alexi Gladstone ALEXI@VIRGINIA.EDU
Kevin Chung KMC5RFU@VIRGINIA.EDU
Sidhardh Burre SSB3VK@VIRGINIA.EDU

Abstract

(Alexi Gladstone)
Convolutions Neural Networks (CNNs) have revolutionized the field of Computer Vision, enabling the training of highly accurate architectures with few parameters. In the context of CNNs, a recent trend has emerged of training highly efficient architectures capable of running on smaller devices. This has high potential for impact in the medical domain, where hardware is often very limited. In this work, we investigate a recently successful efficient architecture called MixConv. MixConv is focused on allowing Convolutional architectures to capture features with different kernel sizes, as to improve computational efficiency and performance. We do comprehensive experimentation regarding different MixConv architectures and its affect on the performance on an image classification task for classifying healthy brains versus those with Alzheimer's disease. The results show that carefully tuning the MixConv architecture can maintain or improve performance while increasing parameter efficiency.

Keywords: Convolution, Efficiency, Medical, Brain, Vision

1. Introduction (Kevin Chung, Sidhardh Burre)

In the world of CNNs, depthwise convolutions are becoming ever more popular. Models such as MobileNet (Howard et al., 2017), ShuffleNet (Zhang et al., 2017), NASNet (Zoph et al., 2018), AmoebaNet (Real et al., 2019), MnasNet (Tan et al., 2019), and EfficientNet (Tan and Le, 2020) have utilized depthwise convolutional layers to improve the accuracy and efficiency when compared to normal ConvNets. Depthwise convolutional layers apply the kernel to each channel individually, reducing the cost of a convolution by a factor equal to the number of channels. These layers are able to improve many CNNs, but when constructing a model with depthwise convolutions, however, changing the kernel size is often overlooked. The MixConv paper looks to further expand on depthwise convolutions by exploring changing kernels sizes in a depthwise convolution.

Conventional kernel size in depthwise convolutions is to use a 3x3 kernel. There has been research that shows some advantage to 5x5 and 7x7 kernels, as in some cases they improve model accuracy and efficiency. Theoretically, it would make sense for a larger kernel size to improve the accuracy of a model, since a larger kernel could technically accomplish the same thing as a smaller kernel, but with more context. Larger kernels are also able to capture larger features of a tensor than smaller kernels. However, it may not always be the case that larger kernel size means higher accuracy.

As shown in Figure 1, the number of parameters in a model increase with the increase in kernel size. In the MobileNetV1, increasing the kernel size to 7x7 improved accuracy, but then the accuracy steeply dropped off after increasing the kernel size further. In the

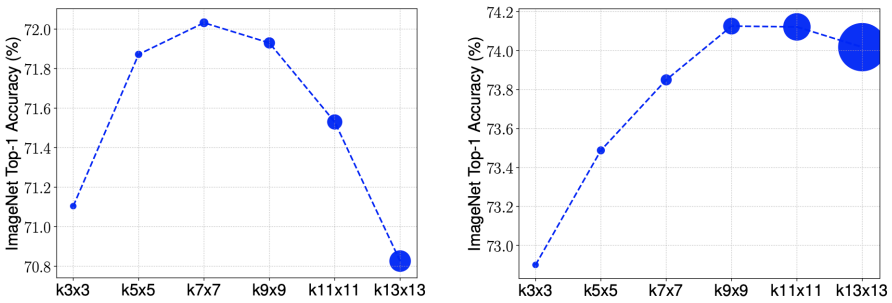

Figure 1: Accuracy vs. kernel size - Each point represents a variant of MobileNetV1 and MobileNetV2 using the corresponding kernel size. Larger points indicate more parameters.

MobileNetV2, there were acccuracy improvements up to the 9x9 kernel size, and a small dropoff in accuracy afterwards. This experiment suggests that increasing the kernel size is not always good. One explanation for the dropoff in accuracy is that larger kernels start to approach a fully-connected network, which is known to perform significantly worse than CNNs.

Smaller kernels are able to capture low-resolution patterns with better accuracy and efficiency and larger kernels are able to capture the higher-resolution patterns in the tensors. MixConv is a mixed depthwise convolution, where different channels have different kernel sizes. By doing this, MixConv layers are able to take advantage of the benefits of both small and large kernel sizes.

1.1. Problem Description

Tan and Le, the MixConv researchers, have already shown MixConv to be effective when replacing convolutions layers in MobileNet and in both ImageNet classification and COCO object detection. They also proposed a MixNet architecture, which is a CNN built using many MixConv layers. When the MixConv paper was released, MixNet-L achieved the state-of-the-art top-1 accuracy of 78.9%.

We would like to further test the performance of different MixNets on a dataset of brain scans with healthy brains and brains with Alzheimer's disease. Alzheimer's disease is one of the most prevalent brain diseases that impacts memory and other important mental functions, so detection accuracy could be very impactful to healthcare and medical diagnoses.

1.2. Related Work

The MixConv paper draws on three primary categories of related works. Each of these related works serves as a baseline or inspiration for MixConv to improve off of.

The first set of related work is the work dedicate to Efficient ConvNets. ConvNets was originally introduced in LeCun's seminal paper (LeCun et al., 1989). The original design had a mere 60 K parameters and a relatively simple architecture as seen in Figure 2.

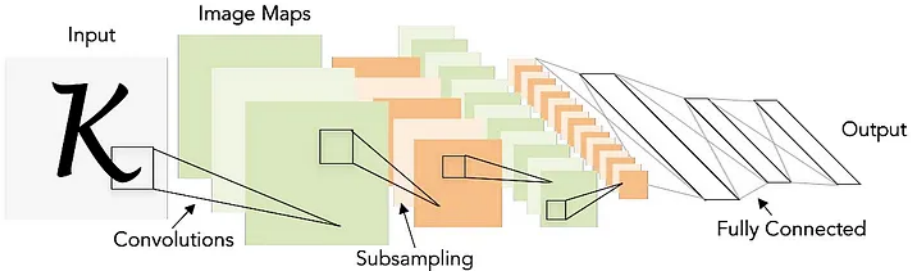

Figure 2: Architecture of LeNet

Since then, continuous and steady progress has been made on improving the ConvNet in the areas of efficiency, size, and accuracy. A significant driving force behind improvements in ConvNet can be attributed to the release of the ImageNet dataset and the "ImageNet large scale visual recognition challenge (ILSVRC)."

Figure 3 is a display of the year over year Classification Results of the ILSVRC competition.

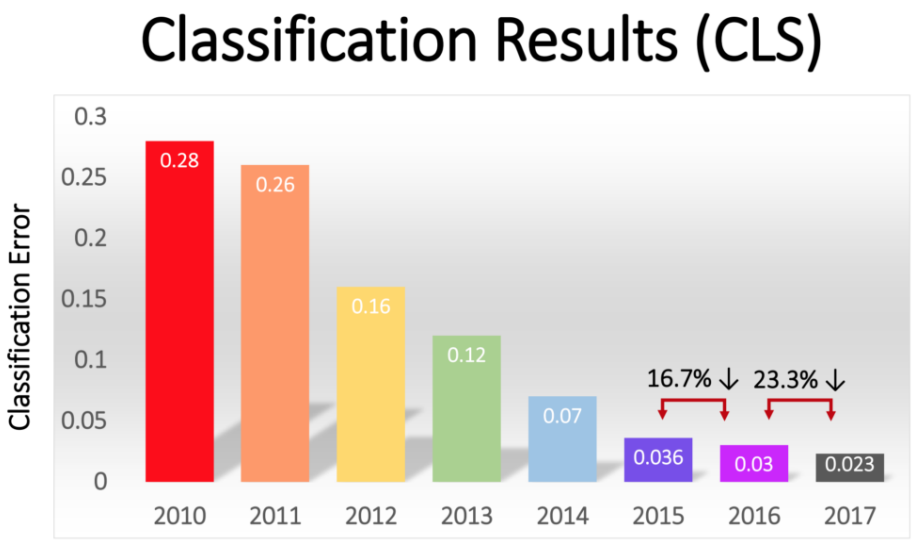

Figure 3: Classification Errors of ILSVRC Winners

The first CNN-based winner of the ILSVRC was AlexNet (Krizhevsky et al., 2012). This was prominent both due to the significant improvement over the incumbent as well as the markedly different architecture. This new model had nearly 60 million parameters and made liberal use of convolutions.

Successive ILSVRC winners have only displayed successive improvements in classification but have also expressed significant increases in model size, as seen in Figure 4.

This seemingly exponential growth in model size poses serious implications for the amount of data required for model training, the computational power required to train the model, as well as the hardware requirements for running the model. The final consid-

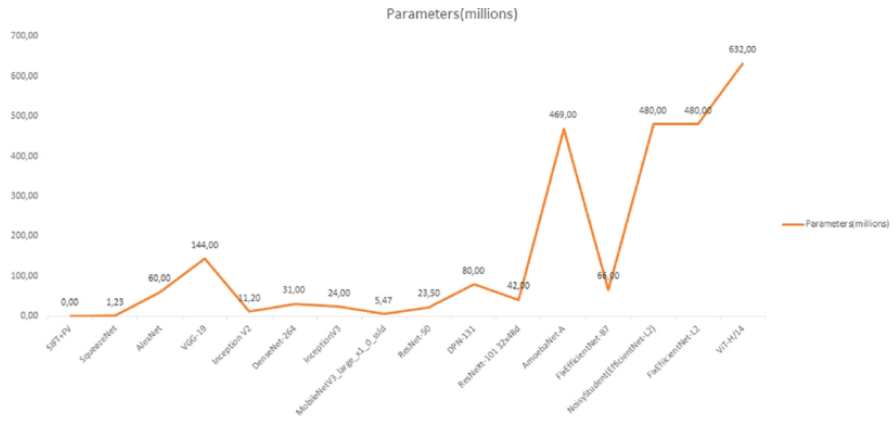

Figure 4: Parameters of ILSVRC Winners

eration is of particular note in mobile and distributed settings. With these considerations in mind, numerous improvements have been made in improving the efficiency of mobile ConvNets.

Efficient ConvNets are a new class of ConvNets that place great emphases on improving ConvNet. These improvements come from more efficient convolutional operations, Bottleneck layers, more efficient architectures, and depthwise convolutions. Of particular importance are the improvements in depthwise convolutions, a key innovation on which MixConv is based off. Figure 5 displays two different types of convolution. In a traditional convolution, all layers of an image are convoluted simultaneously, an operation that scales with the cube of the kernel size. Depthwise convolution, on the other hand, decomposes the image into its constituent layers and computes the convolution on each layer individually. This is an operation that scales with the number of layers and the square of the kernel size, a far more efficient operation.

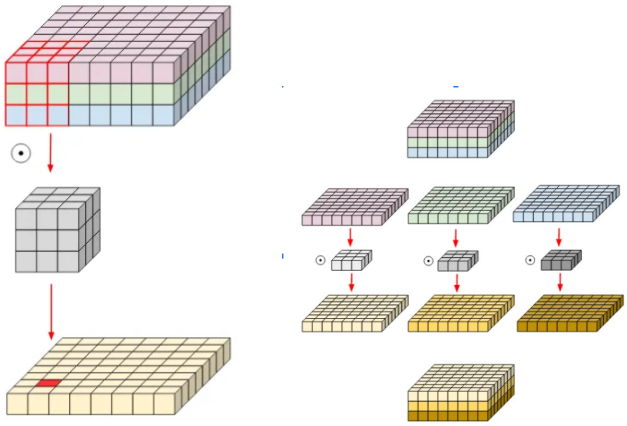

Figure 5: Traditional Convolution (left), Depthwise Convolutions (right)

The next source of inspiration from MixConv's architecture originates from the concept of multi-branch ConvNets. This concept is featured prominently in Inceptions (Szegedy

et al., 2016), ResNeXt (Xie et al., 2017), and NASNet (Zoph et al., 2017). A key characteristic of this feature is the existence of multiple branches in a single layer. This enables the performance of multiple distinct operations in a single layer. As seen in Figure 6, a single layer can conduct multiple different operations. Further, it can employ different *operations* as evidenced by the Pooling operation.

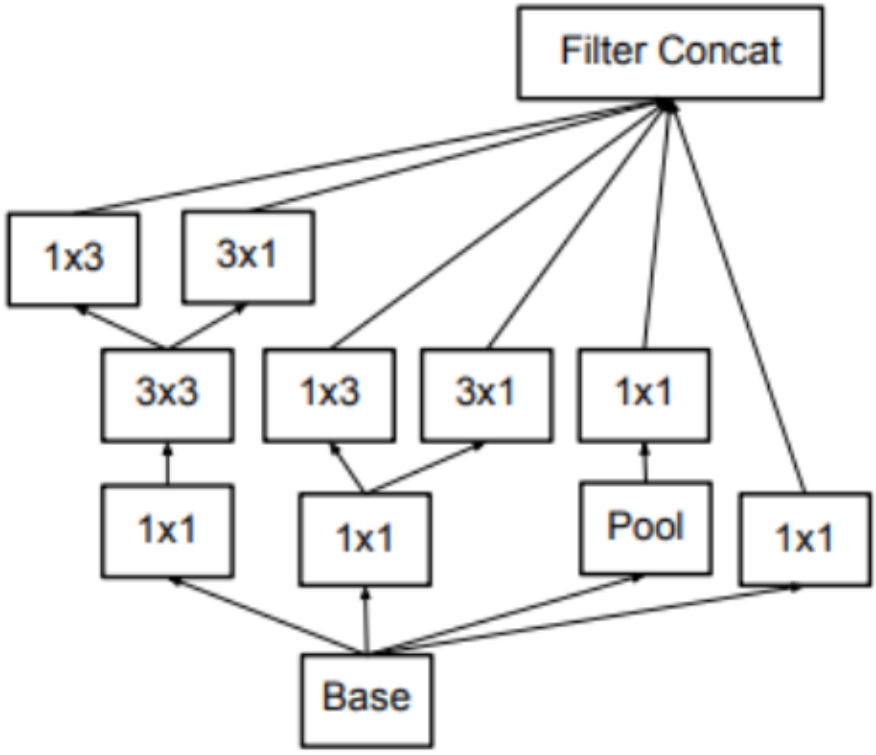

Figure 6: Multi-Branch Architectures

However, these branching architectures impose significant changes on the original architecture. The complete macro-architecture of the neural network changes to adopt a branching pattern. MixConv departs from this concept by instead creating a drop-in layer with incorporated branching to obtain the benefits of interleaving different types of operations while enabling easily use of complex operations.

The final source of inspiration originates from research focused on hyperparameter turning. Recent research into reinforcement learning has demonstrated significant improvements in optimizing an objective function over even hand-tuned optimizations. This class of research, as applied to finding the optimal neural architecture is termed "neural architecture search" (Zoph et al., 2017). Due to the potential complexity posed by MixConv's unique architecture the authors employ neural architecture search similar to that in (Wu et al., 2018)

1.3. Challenge

At the forefront of the difficulties is the technological aspect of neural image classification. These models, while powerful, require vast amounts of data to learn effectively. However, the availability of high-quality, annotated neuroimaging data is limited due to the cost and complexity of acquiring such data. Moreover, Alzheimer's disease manifests in subtle and varied ways in the brain, making it a challenge for even the most advanced algorithms to discern these patterns accurately.

Alzheimer's disease presents a range of biological complexities that further complicate classification efforts. The disease progresses in stages, with early stages often showing minimal or no easily discernible changes in the brain. This subtlety means that distinguishing between a healthy individual and one in the early stages of Alzheimer's can be extremely challenging. Moreover, the disease's progression and manifestation can vary greatly among individuals, adding another layer of complexity to the classification task.

2. Methodology (Kevin Chung, Alexi Gladstone, Sidhardh Burre)

In this section, we will explain the methodology behind the MixConv layers and explain how we will perform our experiments with the Alzheimer's dataset.

2.1. MixConv

Let's start with the normal equation for a depthwise convolution. Let $X^{(h,w,c)}$ denote a tensor of shape (h, w, c) where h and w are the height and width, and c is the number of channels in the tensor. Let $W^{(k,k,c,m)}$ denote a depthwise convolutional kernel where kxk is the kernel size, and m is the channel multiplier. This leads us to the depthwise convolution equation:

$$Y_{x,y,z} = \sum_{-\frac{k}{2} \leq i \leq \frac{k}{2}, -\frac{k}{2} \leq j \leq \frac{k}{2}} X_{x+i,y+j,z/m} \cdot W_{i,j,z}, \qquad \forall z = 1, ..., m \cdot c \tag{1}$$

For MixConv, the tensor is partitioned into g groups on the channel dimension. This can be represented as a vector of virtual tensors, $< \hat{X}^{(h,w,c_1)}, ..., \hat{X}^{(h,w,c_g)} >$, where $c_1 + c_2 + ... + c_g = c$. The convolutional kernels also get partitioned, $< \hat{W}^{(h_1,k_1,c_1,m)}, ..., \hat{W}^{(h_g,w_g,c_g,m)} >$. For the nth group of virtual tensor and kernel, the virtual output tensor is calculated as:

$$\hat{Y}^n_{x,y,z} = \sum_{-\frac{k_n}{2} \leq i \leq \frac{k_n}{2}, -\frac{k_n}{2} \leq j \leq \frac{k_n}{2}} \hat{X}^n_{x+i,y+j,z/m} \cdot \hat{W}^n_{i,j,z}, \qquad \forall z = 1, ..., m \cdot c_n \tag{2}$$

Afterwards, the final output tensor is calculated by concatenating the virtual output tensors:

$$Y_{x,y,z_o} = Concat(\hat{Y}^1_{x,y,z_1}, ..., \hat{Y}^g_{x,y,z_g}) \tag{3}$$

The final output channel size is $z_o = z_1 + ... + z_g = m \cdot c$.

2.2. Alzheimer's dataset

The Alzheimer's dataset is composed of 856 .raw images of diseased and healthy brains of size 256 by 256 pixels. This dataset was further broken down as indicated in Table 1.

	Train	Test
Health	91	12
Diseased	287	38

Table 1: Train and Test split of preprocessed dataset

The dataset comes with two splits—train and test, but without a validation set. Therefore, the training data was randomly split into a training and validation dataset.

3. Experiment (Alexi Gladstone)

For all experiments we utilized a container, as to manage package dependencies and increase reproducibility. Additionally, we used the Hugging Face Pytorch Image Models Repo to start (Wightman, 2019). We used the following nonexhaustive list of hyperparameters for experimentation: batch size = 32, epochs = 20, decay-epochs = 2.4, warm-up learning rate = $1e-6$.

For our first experiment, we do a search over 5 different learning rates—$0.3, 0.03$, $0.003, 0.0003$, and 0.00003. Plots showing the validation (evaluation) loss for these 5 learning rates are shown in Fig. 7. The results show that a learning rate of 0.03 is optimal, so we use this for all the below experimentation.

Next, having found a learning rate that enabled smooth training, we initially experimented with 8 different kernel size combinations for the mixnet architecture: $(3, 5, 7)$, $(5, 7, 9)$, $(7, 9, 11)$, $(9, 11, 13)$, $(11, 13, 15)$, $(13, 15, 17)$, $(15, 17, 19)$, and $(17, 19, 21)$. The goal of this experiment was to demonstrate how different kernel combination saffect the accuracy. The validation loss for these different configurations are shown in Fig. 7. Additionally, the resulting test set performances are shown in Table 2. We also check the total number of parameters for varying combinations of kernel sizes in these experiments and found that increasing the sum of kernel sizes increases the total number of parameters (and training time). This can also be seen in Table 2. This is expected as increasing the kernel size increases the total number of parameters.

Additionally, due to the consistent performance of the above architectures, we also experimented with a weighted loss to manage the inherent imbalance in classes (Diseased having around 3x as many samples as healthy). Thus, the results in Table 3 show these, with the network still achieving a maximum performance of 75.51%.

4. Discussion (Kevin Chung and Alexi Gladstone)

4.1. Results

The best test set accuracy we were able to get was 75.51%. This was pretty consistently scored across kernel sizes. We believe this was a commonly scored result due to it being the percentage of the test set corresponding to the majority class (diseased brains). Therefore,

Figure 7: Learning Rate vs. Evaluation Loss (left), Kernel Combination vs. Unweighted Evaluation Loss (center), Kernel Combination vs. Weighted Evaluation Loss (right)

Kernel Sizes	Parameter Count	Test Accuracy
3.5.7	2600680	75.51%
5.7.9	2687944	75.51%
7.9.11	2804296	75.51%
9.11.13	2949736	75.51%
11.13.15	3124264	75.51%
13.15.17	3327880	75.51%
15.17.19	3560584	75.51%
17.19.21	3822376	75.51%

Table 2: Kernel Size, Parameter Count, and Test Set Accuracy using a weighted loss.

Kernel Sizes	Test Accuracy
3.5.7	75.51%
5.7.9	24.49%
7.9.11	75.51%
9.11.13	24.49%
11.13.15	75.51%
13.15.17	75.51%
15.17.19	24.49%
17.19.21	75.51%

Table 3: Kernel Size, Parameter Count, and Test Set Accuracy using a weighted loss.

it is likely that even though the model's loss decreased during training it was unable to distinguish very well between brains with/without Alzheimer's due to the challenging nature of the task and the very limited dataset. This makes sense as even we could not distinguish between health/diseased brains, and we could be considered experts. Additionally, having only hundreds of data samples to train on, the model was simply unable to distinguish well and resulted to what maximized the likelihood which was picking the majority class.

4.2. Limitations

One of the most important limitations of MixConv is potential overfitting. Even though a MixConv layer will be more efficient and have less parameters than a vanilla depthwise convolutional layer, it still does not address the overfitting problem that many CNNs expe-

rience. MixConv also still has more parameters and is less computationally efficient than a standard convolutional layer.

Another important issue to address when using a neural network with MixConv layers is the increase in hyperparameter tuning. Rather than having to establish a single kernel size for a layer, one would need to establish the range of convolutional kernel sizes. The MixConv researchers did propose using a neural architecture search to optimize the CNN structure, but this just adds more time to the overall fitting process.

For our experiment, we were limited by the small size of our dataset. The training dataset has 260 examples with Alzheimer's and 82 examples without. The validation set only has 28 examples with Alzheimer's and 9 without. This is not nearly enough to create a model that will generalize well. The distribution of the data is also not ideal, with there being over 3 times as many diseased examples as health examples in our dataset. The diversity of the data is also subpar, given the low number of examples. Due to the small amount of data, we also were only able to achieve around a 75% accuracy. We hypothesize that if we had more data, the model would converge more and have a higher classification accuracy. We also analyzed the dataset visually, and could not pick out the difference between the images with Alzheimer's and those without.

4.3. Future work

The MixConv researchers posed a questions in their paper of which kernel size is the best. MixConv is a good way to generalize kernel sizes, but more research could be done into when you want to have a larger or smaller kernel in your CNN. Another future research direction is to test out MixConv with larger models, rather than just mobile models. Since MixConv was able to improve accuracy and efficiency in mobile nets, it would probably also improve accuracy and efficiency in much larger models that use depthwise convolutions. We are also interested to see if mixing kernels sizes is applicable in other domains where convolutions are used, such as audio.

5. Conclusion (Kevin Chung, Alexi Gladstone)

MixConv had a very strong contribution to the world of computer vision and CNNs. Mixing kernel sizes is a novel idea that allows us leverage advantages from both smaller *and* larger kernels simultaneously. Future work could focus on understanding how and why different kernel approaches work theoretically, especially in the medical domain. Additionally, gathering a more comprehensive dataset, with more samples as well as less of a class imbalance, could improve performance.

References

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, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks, 2012. URL https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition, 1989.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2019.

Christian Szegedy, Sergey Ioffe, and Vincent Vanhoucke. Inception-v4, inception-resnet and the impact of residual connections on learning, 2016. URL http://arxiv.org/abs/1602.07261.

Mingxing Tan and Quoc V. Le. Efficientnet: Rethinking model scaling for convolutional neural networks, 2020.

Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V. Le. Mnasnet: Platform-aware neural architecture search for mobile, 2019.

Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019.

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. *CoRR*, abs/1812.03443, 2018. URL http://arxiv.org/abs/1812.03443.

Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5987–5995, 2017. doi: 10.1109/CVPR.2017.634.

Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices, 2017.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2017. URL http://arxiv.org/abs/1707.07012.

Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2018.