MixNet: Mixed Depthwise Convolutional Kernels

Mingxing Tan
tanmingxing@google.com
Quoc V. Le
qvl@google.com
Google Brain
Mountain View, CA, USA

Abstract

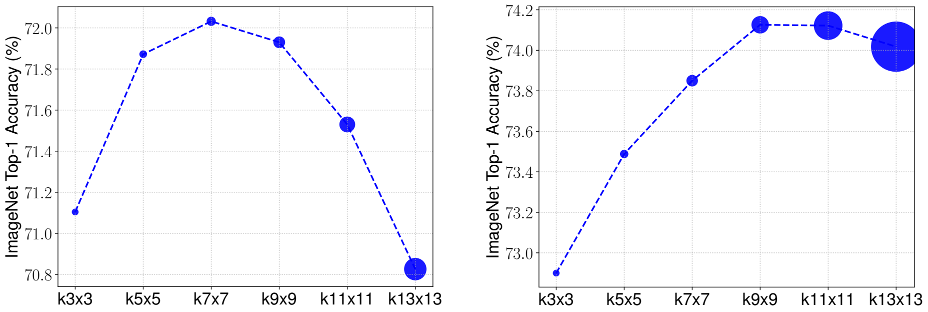
Depthwise convolution is becoming increasingly popular in modern efficient ConvNets, but its kernel size is often overlooked. In this paper, we systematically study the impact of different kernel sizes, and observe that combining the benefits of multiple kernel sizes can lead to better accuracy and efficiency. Based on this observation, we propose a new mixed depthwise convolution (**MDConv**), which naturally mixes up multiple kernel sizes in a single convolution. As a simple drop-in replacement of vanilla depthwise convolution, our MDConv improves the accuracy and efficiency for existing MobileNets on both ImageNet classification and COCO object detection.

By integrating MDConv into AutoML search space, we have further developed a new family of models, named as **MixNets**, which significantly outperform previous models including MobileNetV2 [19] (ImageNet top-1 accuracy +4.2%), ShuffleNetV2 [15] (+3.5%), MnasNet [25] (+1.3%), ProxylessNAS [2] (+2.2%), and FBNet [26] (+2.0%). In particular, our MixNet-L achieves a new state-of-the-art 78.9% ImageNet top-1 accuracy under typical mobile settings (<600M FLOPS). Code is at `https://github.com/tensorflow/tpu/tree/master/models/official/mnasnet/mixnet`.

1 Introduction

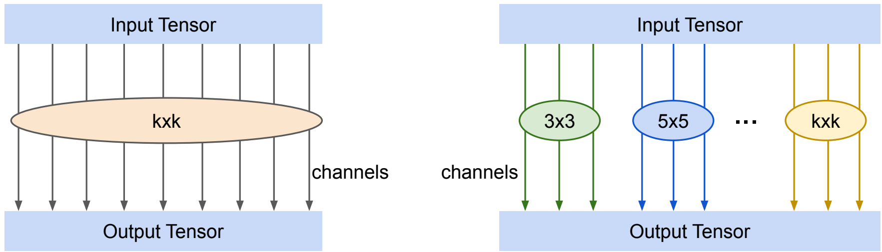
Convolutional neural networks (ConvNets) have been widely used in image classification, detection, segmentation, and many other applications. A recent trend in ConvNets design is to improve both accuracy and efficiency. Following this trend, depthwise convolutions are becoming increasingly more popular in modern ConvNets, such as MobileNets [6, 19], ShuffleNets [15, 29], NASNet [31], AmoebaNet [17], MnasNet [25], and EfficientNet [24]. Unlike regular convolution, depthwise convolutional kernels are applied to each individual channel separately, thus reducing the computational cost by a factor of C, where C is the number of channels. While designing ConvNets with depthwise convolutional kernels, an important but often overlooked factor is kernel size. Although conventional practice is to simply use 3x3 kernels [3, 6, 15, 19, 29, 31], recent research results have shown larger kernel sizes such as 5x5 kernels [25] and 7x7 kernels [2] can potentially improve model accuracy and efficiency.

In this paper, we revisit the fundamental question: *do larger kernels always achieve higher accuracy?* Since first observed in AlexNet [11], it has been well-known that each convolutional kernel is responsible to capture a local image pattern, which could be edges in early stages and objects in later stages. Large kernels tend to capture high-resolution patterns



(a) MobileNetV1 (b) MobileNetV2

Figure 1: **Accuracy vs kernel sizes** – Each point represents a model variant of MobileNet V1[8] and V2 [19], where model size is represented by point size. Larger kernels lead to more parameters, but the accuracy actually drops down when kernel size is larger than 9x9.



(a) Vanilla Depthwise Convolution (b) Our proposed MDConv

Figure 2: **Mixed depthwise convolution (MDConv)** – Unlike vanilla depthwise convolution that applies a single kernel to all channels, MDConv partitions channels into groups and apply different kernel size to each group.

with more details at the cost of more parameters and computations, but do they always improve accuracy? To answer this question, we systematically study the impact of kernel sizes based on MobileNets [8, 19]. Figure 1 shows the results. As expected, larger kernel sizes significantly increase the model size with more parameters; however, model accuracy first goes up from 3x3 to 7x7, but then drops down quickly when the kernel size is larger than 9x9, suggesting very large kernel sizes can potentially hurt both accuracy and efficiency. In fact, this observation aligns to the very first intuition of ConvNets: in the extreme case that kernel size is equal to the input resolution, a ConvNet simply becomes a fully-connected network, which is known to be inferior [7]. This study suggests the limitations of single kernel size: we need both large kernels to capture high-resolution patterns and small kernels to capture low-resolution patterns for better model accuracy and efficiency.

Based on this observation, we propose a *mixed depthwise convolution (MDConv)*, which mixes up different kernel sizes in a single convolution op, such that it can easily capture different patterns with various resolutions. Figure 2 shows the structure of MDConv, which partitions channels into multiple groups and apply different kernel sizes to each group of channels. We show that our MDConv is a simple drop-in replacement of vanilla depthwise convolution, but it can significantly improve MobileNets accuracy and efficiency on both ImageNet classification and COCO object detection.

To further demonstrate the effectiveness of our MDConv, we leverage neural architecture search [25] to develop a new family of models named as *MixNets*. Experimental results show our MixNet models significantly outperform all previous mobile ConvNets, such as ShuffleNets [15, 29], MnasNet [25], FBNet [26], and ProxylessNAS [2]. In particular, our medium-size MixNet-M achieves the same 77.0% ImageNet top-1 accuracy, while using 12x fewer parameters and 31x fewer FLOPS than ResNet-152 [4].

2 Related Work

Efficient ConvNets: In recent years, significant efforts have been spent on improving ConvNet efficiency, from more efficient convolutional operations [3, 5, 8], bottleneck layers [19, 27], to more efficient architectures [2, 25, 26]. In particular, depthwise convolution has been increasingly popular in all mobile-size ConvNets, such as MobileNets [5, 19], ShuffleNets[15, 29], MnasNet [25], and beyond [3, 17, 31]. Recently, EfficientNet [24] even achieves both state-of-the-art ImageNet accuracy and ten-fold better efficiency by extensively using depthwise and pointwise convolutions. Unlike regular convolution, depthwise convolution performs convolutional kernels for each channel separately, thus reducing parameter size and computational cost. Our proposed MDConv generalizes the concept of depthwise convolution, and can be considered as a drop-in replacement of vanilla depthwise convolution.

Multi-Scale Networks and Features: Our idea shares a lot of similarities to prior multi-branch ConvNets, such as Inceptions [21, 23], Inception-ResNet [22], ResNeXt [27], and NASNet [31]. By using multiple branches in each layer, these ConvNets are able to utilize different operations (such as convolution and pooling) in a single layer. Similarly, there are also many prior work on combining multi-scale feature maps from different layers, such as DenseNet [6, 7] and feature pyramid network [12]. However, unlike these prior works that mostly focus on changing the macro-architecture of neural networks in order to utilize different convolutional ops, our work aims to design a drop-in replacement of a single depthwise convolution, with the goal of easily utilizing different kernel sizes without changing the network structure.

Neural Architecture Search: Recently, neural architecture search [13, 14, 25, 30, 31] has achieved better performance than hand-crafted models by automating the design process and learning better design choices. Since our MDConv is a flexible operation with many possible design choices, we employ existing architecture search methods similar to [2, 25, 26] to develop a new family of MixNets by adding our MDConv into the search space.

3 MDConv

The main idea of MDConv is to mix up multiple kernels with different sizes in a single depthwise convolution op, such that it can easily capture different types of patterns from input images. In this section, we will discuss the feature map and design choices for MDConv.

3.1 MDConv Feature Map

We start from the vanilla depthwise convolution. Let $X^{(h,w,c)}$ denotes the input tensor with shape (h,w,c), where c is the spatial height, w is the spatial width, and c is the channel size.

Let $W^{(k,k,c,m)}$ denotes a depthwise convolutional kernel, where $k \times k$ is the kernel size, c is the input channel size, and m is the channel multiplier. For simplicity, here we assume kernel width and height are the same k, but it is straightforward to generalize to cases where kernel width and height are different.s The output tensor $Y^{(h,w,c \cdot m)}$ would have the same spatial shape (h,w) and multiplied output channel size $m \cdot c$, with each output feature map value calculated as:

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

Unlike vanilla depthwise convolution, MDConv partitions channels into groups and applies different kernel sizes to each group, as shown in Figure 2. More concretely, the input tensor is partitioned into g groups of virtual tensors $< \hat{X}^{(h,w,c_1)}, ..., \hat{X}^{(h,w,c_g)} >$, where all virtual tensors $\hat{X}$ have the same spatial height h and width w, and their total channel size is equal to the original input tensor: $c_1 + c_2 + ... + c_g = c$. Similarly, we also partition the convolutional kernel into g groups of virtual kernels $< \hat{W}^{(k_1,k_1,c_1,m)}, ..., \hat{W}^{(k_g,k_g,c_g,m)} >$. For $t-$th group of virtual input tensor and kernel, the corresponding virtual output is calculated as:

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

The final output tensor is a concatenation of all virtual output tensors $< \hat{Y}^1_{x,y,z_1}, ..., \hat{Y}^g_{x,y,z_g} >$:

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

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

Figure 3 shows a simple demo of Tensorflow python implementation for MDConv. On certain platforms, MDConv could be implemented as a single op and optimized with group convolution. Nevertheless, as shown in the figure, MDConv can be considered as a simple drop-in replacement of vanilla depthwise convolution.

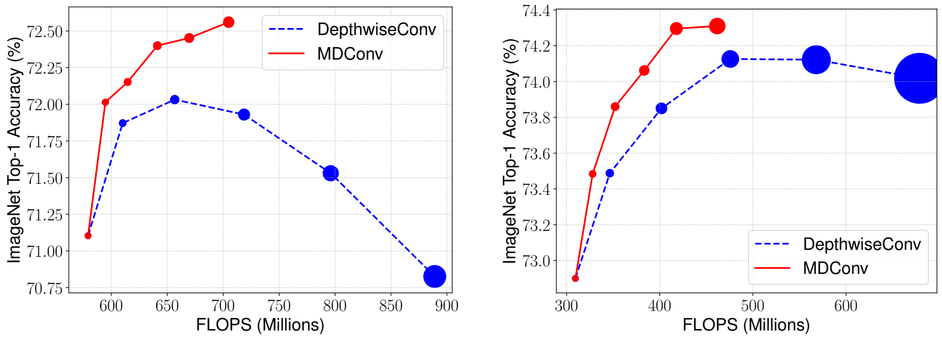
```
def mdconv(x, filters, **args):
    # x: input features with shape [N,H,W,C]
    # filters: a list of filters with shape [K_i, K_i,
        C_i, M_i] for i−th group.
    G = len(filters) # number of groups.
    y = []
    for xi, fi in zip(tf.split(x, G, axis=−1), filters):
        y.append(tf.nn.depthwise_conv2d(xi, fi, **args))
    return tf.concat(y, axis=−1)
```

Figure 3: A demo of TensorFlow MDConv.

3.2 MDConv Design Choices

MDConv is a flexible convolutional op with several design choices:

Group Size g: It determines how many different types of kernels to use for a single input tensor. In the extreme case of $g = 1$, a MDConv becomes equivalent to a vanilla depthwise convolution. In our experiments, we find $g = 4$ is generally a safe choice for MobileNets, but with the help of neural architecture search, we find it can further benefit the model efficiency and accuracy with a variety of group sizes from 1 to 5.



(a) MobileNetV1 (b) MobileNetV2

Figure 4: **MDConv performance on ImageNet** – Each point denotes a model with kernel size from 3x3 to 13x13, same as Figure 1. `MDConv` is smaller, faster, and achieves higher accuracy than vanilla depthwise convolutions.

Kernel Size Per Group: In theory, each group can have arbitrary kernel size. However, if two groups have the same kernel size, then it is equivalent to merge these two groups into a single group, so we restrict each group has different kernel size. Furthermore, since small kernel sizes generally have less parameters and FLOPS, we restrict kernel size always starts from 3x3, and monotonically increases by 2 per group. In other words, group i always has kernel size $2i+1$. For example, a 4-group MDConv always uses kernel sizes {3x3, 5x5, 7x7, 9x9}. With this restriction, the kernel size for each group is predefined for any group size g, thus simplifying our design process.

Channel Size Per Group: In this paper, we mainly consider two channel partition methods: (1) Equal partition: each group will have the same number of filters; (2) Exponential partition: the i-th group will have about 2^{-i} portion of total channels. For example, given a 4-group MDConv with total filter size 32, the equal partition will divide the channels into (8, 8, 8, 8), while the exponential partition will divide the channels into (16, 8, 4, 4).

Dilated Convolution: Since large kernels need more parameters and computations, an alternative is to use dilated convolution [28], which can increase receptive field without extra parameters and computations. However, as shown in our ablation study in Section 3.4, dilated convolutions usually have inferior accuracy than large kernel sizes.

3.3 MDConv Performance on MobileNets

Since MDConv is a simple drop-in replacement of vanilla depthwise convolution, we evaluate its performance on classification and detection tasks with existing MobileNets [5, 19].

ImageNet Classification Performance: Figure 4 shows the performance of MDConv on ImageNet classification [18]. Based on MobileNet V1 and V2, we replace all original 3x3 depthwise convolutional kernels with larger kernels or MDConv kernels. Notably, MDConv always starts with 3x3 kernel size and then monotonically increases by 2 per group, so the rightmost point for `MDConv` in the figure has six groups of filters with kernel size {3x3, 5x5, 7x7, 9x9, 11x11, 13x13}. In this figure, we observe: (1) MDConv generally uses much less parameters and FLOPS, but its accuracy is similar or better than vanilla depthwise convolution, suggesting mixing different kernels can improve both efficiency and accuracy;

Network	MobileNetV1 [6] #Params	#FLOPS	*mAP*	MobileNetV2 [19] #Params	#FLOPS	*mAP*
baseline3x3	5.12M	1.31B	21.7	4.35M	0.79B	21.5
depthwise5x5	5.20M	1.38B	22.3	4.47M	0.87B	22.1
mdconv 35 (ours)	**5.16M**	**1.35B**	**22.2**	**4.41M**	**0.83B**	**22.1**
depthwise7x7	5.32M	1.47B	21.8	4.64M	0.98B	21.2
mdconv 357 (ours)	**5.22M**	**1.39B**	**22.4**	**4.49M**	**0.88B**	**22.3**

Table 1: **Performance comparison on COCO object detection.**

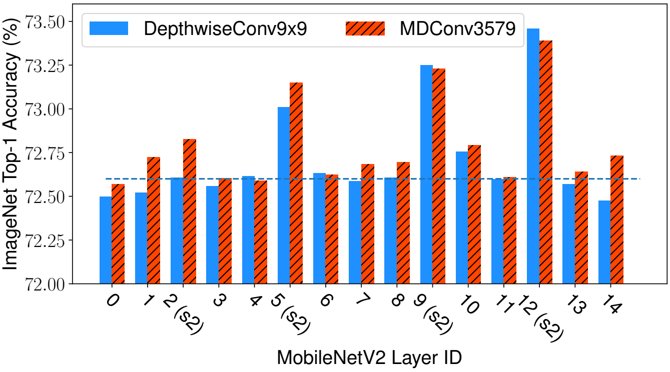


Figure 5: **Per-layer impact of kernel size** – `s2` denotes stride 2, while others have stride 1.

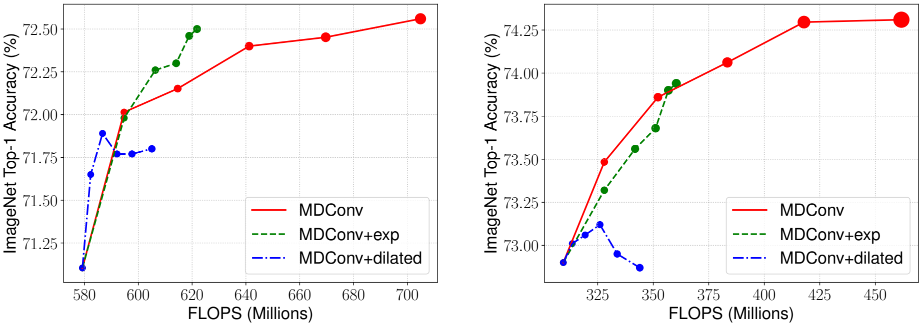
(2) In contrast to vanilla depthwise convolution that suffers from accuracy degradation with larger kernels, as shown in Figure 1, MDConv is much less sensitive to very large kernels, suggesting mixing different kernels can achieve more stable accuracy for large kernel sizes.

COCO Detection Performance: We have also evaluated our MDConv on COCO object detection based on MobileNets. Table 1 shows the performance comparison, where our MDConv consistently achieves better efficiency and accuracy than vanilla depthwise convolution. In particular, compared to the vanilla depthwise7x7, our MDConv357 (with 3 groups of kernels {3x3, 5x5, 7x7}) achieves 0.6% higher mAP on MobileNetV1 and 1.1% higher mAP on MobileNetV2 using fewer parameters and FLOPS.

3.4 Ablation Study

To better understand MDConv, we provide a few ablation studies:

MDConv for Single Layer: In addition of applying MDConv to the whole network, Figure 5 shows the per-layer performance on MobileNetV2. We replace one of the 15 layers with either (1) vanilla `DepthwiseConv9x9` with kernel size 9x9; or (2) `MDConv3579` with 4 groups of kernels: {3x3, 5x5, 7x7, 9x9}. As shown in the figure, large kernel size has different impact on different layers: for most of layers, the accuracy doesn't change much, but for certain layers with stride 2, a larger kernel can significantly improve the accuracy. Notably, although `MDConv3579` uses only half parameters and FLOPS than the vanilla `DepthwiseConv9x9`, our MDConv achieves similar or slightly better performance for most of the layers.



(a) MobileNetV1 (b) MobileNetV2

Figure 6: **Study for exponential channel partition (+exp) and dilated kernels (+dilated).**

Channel Partition Methods: Figure 6 compares the two channel partition methods: equal partition (MDConv) and exponential partition (MDConv+exp). As expected, exponential partition requires less parameters and FLOPS for the same kernel size, by assigning more channels to smaller kernels. Our empirical study shows exponential channel partition only performs slightly better than equal partition on MobileNetV1, but there is no clear winner if considering both MobileNet V1 and V2. A possible limitation of exponential partition is that large kernels won't have enough channels to capture high-resolution patterns.

Dilated Convolution: Figure 6 also compares the performance of dilated convolution (denoted as MDConv+dilated). For kernel size KxK, it uses a 3x3 kernel with dilation rate $(K-1)/2$: for example, a 9x9 kernel will be replaced by a 3x3 kernel with dilation rate 4. Notably, since Tensorflow dilated convolution is not compatible with stride 2, we only use dilated convolutions for a layer if its stride is 1. As shown in the figure, dilated convolution has reasonable performance for small kernels, but the accuracy drops quickly for large kernels. Our hypothesis is that when dilation rate is big for large kernels, a dilated convolution will skip a lot of local information, which would hurt the accuracy.

4 MixNet

To further demonstrate the effectiveness of MDConv, we leverage recent progress in neural architecture search to develop a new family of MDConv-based models, named as MixNets.

4.1 Architecture Search

Our neural architecture search settings are similar to recent MnasNet [25], FBNet [26], and ProxylessNAS [2], which use MobileNetV2 [19] as the baseline network structure, and search for the best kernel size, expansion ratio, channel size, and other design choices. However, unlike these prior works that use vanilla depthwise convolution as the basic convolutional op, we adopt our proposed MDConv as the search options. Specifically, we have five MDConv candidates with group size $g = 1, \ldots, 5$:

- **3x3**: MDConv with one group of filters ($g = 1$) with kernel size 3x3.
- ...

Model	Type	#Parameters	#FLOPS	Top-1 (%)	Top-5 (%)
MobileNetV1 [5]	manual	4.2M	575M	70.6	89.5
MobileNetV2 [19]	manual	3.4M	300M	72.0	91.0
MobileNetV2 (1.4x)	manual	6.9M	585M	74.7	92.5
ShuffleNetV2 [15]	manual	-	299M	72.6	-
ShuffleNetV2 (2x)	manual	-	597M	75.4	-
ResNet-153 [4]	manual	60M	11B	77.0	93.3
NASNet-A [31]	auto	5.3M	564M	74.0	91.3
DARTS [14]	auto	4.9M	595M	73.1	91
MnasNet-A1 [25]	auto	3.9M	312M	75.2	92.5
MnasNet-A2	auto	4.8M	340M	75.6	92.7
FBNet-A [26]	auto	4.3M	249M	73.0	-
FBNet-C	auto	5.5M	375M	74.9	-
ProxylessNAS [2]	auto	4.1M	320M	74.6	92.2
ProxylessNAS (1.4x)	auto	6.9M	581M	76.7	93.3
MixNet-S	**auto**	**4.1M**	**256M**	**75.8**	**92.8**
MixNet-M	**auto**	**5.0M**	**360M**	**77.0**	**93.3**
MixNet-L	**auto**	**7.3M**	**565M**	**78.9**	**94.2**

Table 2: **MixNet performance results on ImageNet 2012 [18].**

- **3x3, 5x5, 7x7, 9x9, 11x11**: MDConv with five groups of filters ($g = 5$) with kernel size {3x3, 5x5, 7x7, 9x9, 11x11}. Each group has roughly the same number of channels.

In order to simplify the search process, we don't include exponential channel partition or dilated convolutions in our search space, but it is trivial to integrate them in future work.

Similar to recent neural architecture search approaches [2, 25, 26], we directly search on ImageNet train set, and then pick a few top-performing models from search to verify their accuracy on ImageNet validation set and transfer learning datasets.

4.2 MixNet Performance on ImageNet

Table 2 shows the ImageNet performance of MixNets. Here we obtain MixNet-S and M from neural architecture search, and scale up MixNet-M with depth multiplier 1.3 to obtain MixNet-L. All models are trained with the same settings as MnasNet [25].

In general, our MixNets significantly outperform all latest mobile ConvNets: Compared to the latest hand-crafted models, our MixNets improve top-1 accuracy by 4.2% than MobileNetV2 [19] and 3.5% than ShuffleNetV2 [15], under the same FLOPS constraint; Compared to the latest automated models, our MixNets achieve significantly better accuracy than MnasNet (+1.3%), FBNets (+2.0%), ProxylessNAS (+2.2%) under similar FLOPS constraint. In particular, our MixNet-L achieves a new state-of-the-art 78.9% top-1 accuracy under typical mobile FLOPS (<600M) constraint. Compared to the widely used ResNets [4], our MixNet-M achieves the same 77% top-1 accuracy, while using 12x fewer parameters and 31x fewer FLOPS than ResNet-152.

Figure 7 visualizes the ImageNet performance comparison. We observe that recent progresses on neural architecture search have significantly improved model performance [2, 25, 26] than previous hand-crafted mobile ConvNets [15, 19]. However, by introducing a

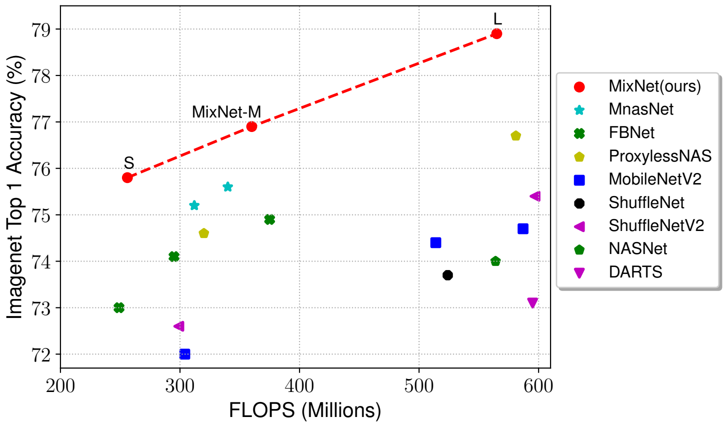


Figure 7: **ImageNet performance comparison.**

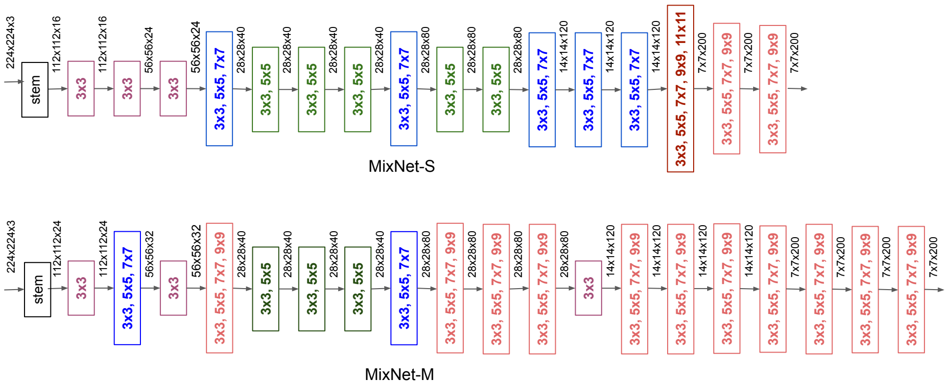


Figure 8: **MixNet architectures** – `MixNet-S` and `MixNet-M` are from Table 2. We mainly highlight MDConv kernel size (e.g. {3x3, 5x5}) and input/output tensor shape.

new type of efficient MDConv, we can further improve model accuracy and efficiency based on the same neural architecture search techniques.

4.3 MixNet Architectures

To understand why our MixNets achieve better accuracy and efficiency, Figure 8 illustrates the network architecture for `MixNet-S` and `MixNet-M` from Table 2. In general, they both use a variety of MDConv with different kernel sizes throughout the network: small kernels are more common in early stage for saving computational cost, while large kernels are more common in later stage for better accuracy. We also observe that the bigger `MixNet-M` tends to use more large kernels and more layers to pursing higher accuracy, with the cost of more parameters and FLOPS. Unlike vanilla depthwise convolutions that suffer from serious accuracy degradation for large kernel sizes (Figure 1), our MixNets are capable of utilizing very large kernels such as 9x9 and 11x11 to capture high-resolution patterns from input images, without hurting model accuracy and efficiency.

Dataset	TrainSize	TestSize	Classes
CIFAR-10 [10]	50,000	10,000	10
CIFAR-100 [10]	50,000	10,000	100
Oxford-IIIT Pets [16]	3,680	3,369	37
Food-101 [1]	75,750	25,250	101

Table 3: **Transfer learning datasets.**

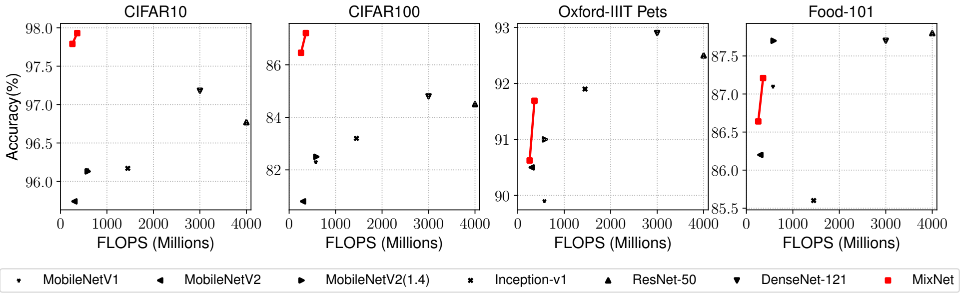


Figure 9: **Transfer learning performance** – `MixNet-S/M` are from Table 2.

4.4 Transfer Learning Performance

We have also evaluated our MixNets on four widely used transfer learning datasets, including CIFAR-10/100 [10], Oxford-IIIT Pets [16] , and Food-101 [1]. Table 3 shows their statistics of train set size, test set size, and number of classes.

Figure 9 compares our `MixNet-S/M` with a list of previous models on transfer learning accuracy and FLOPS. For each model, we first train it from scratch on ImageNet and than finetune all the weights on the target dataset using similar settings as [9]. The accuracy and FLOPS data for MobileNets [5, 19], Inception [20], ResNet [4], DenseNet [7] are from [9]. In general, our MixNets significantly outperform previous models on all these datasets, especially on the most widely used CIFAR-10 and CIFAR-100, suggesting our MixNets also generalize well to transfer learning. In particular, our `MixNet-M` achieves 97.92% accuracy with 3.49M parameters and 352M FLOPS, which is **11.4x** more efficient with 1% higher accuracy than ResNet-50 [4].

5 Conclusions

In this paper, we revisit the impact of kernel size for depthwise convolution, and identify that traditional depthwise convolution suffers from the limitations of single kernel size. To address this issue, we proposes MDConv, which mixes multiple kernels in a single op to take advantage of different kernel sizes. We show that our MDConv is a simple drop-in replacement of vanilla depthwise convolution, and improves the accuracy and efficiency for MobileNets, on both image classification and object detection tasks. Based on our proposed MDConv, we further develop a new family of MixNets using neural architecture search techniques. Experimental results show that our MixNets achieve significantly better accuracy and efficiency than all latest mobile ConvNets on both ImageNet classification and four widely used transfer learning datasets.

References

[1] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. *ECCV*, pages 446–461, 2014.

[2] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *ICLR*, 2019.

[3] François Chollet. Xception: Deep learning with depthwise separable convolutions. *CVPR*, pages 1610–02357, 2017.

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

[5] 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.

[6] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *ICLR*, 2017.

[7] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. *CVPR*, 2017.

[8] 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. *arXiv preprint arXiv:1602.07360*, 2016.

[9] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? *CVPR*, 2018.

[10] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *Technical Report*, 2009.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[12] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. *CVPR*, 2017.

[13] Chenxi Liu, Barret Zoph, Jonathon Shlens, Wei Hua, Li-Jia Li, Li Fei-Fei, Alan Yuille, Jonathan Huang, and Kevin Murphy. Progressive neural architecture search. *ECCV*, 2018.

[14] Hanxiao Liu, Karen Simonyan, and Yiming Yang. DARTS: Differentiable architecture search. *ICLR*, 2019.

[15] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. *ECCV*, 2018.

[16] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. *CVPR*, pages 3498–3505, 2012.

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

[18] 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.

[19] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. *CVPR*, 2018.

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

[21] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *CVPR*, pages 2818–2826, 2016.

[22] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI*, 4:12, 2017.

[23] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI*, 4:12, 2017.

[24] Mingxing Tan and Quoc V Le. *ICML*, 2019.

[25] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. *CVPR*, 2019.

[26] 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. *CVPR*, 2019.

[27] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. *CVPR*, pages 5987–5995, 2017.

[28] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *ICLR*, 2016.

[29] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *CVPR*, 2018.

[30] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *ICLR*, 2017.

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