

# Shift: A Zero FLOP, Zero Parameter Alternative to Spatial Convolutions

Bichen Wu, Alvin Wan\*, Xiangyu Yue\*, Peter Jin, Sicheng Zhao,  
Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, Kurt Keutzer  
UC Berkeley

{bichen, alvinwan, xyyue, phj, schzhao, noah.golmant, amirgh, jegonzal, keutzer}@berkeley.edu

## Abstract

Neural networks rely on convolutions to aggregate spatial information. However, spatial convolutions are expensive in terms of model size and computation, both of which grow quadratically with respect to kernel size. In this paper, we present a parameter-free, FLOP-free “shift” operation as an alternative to spatial convolutions. We fuse shifts and point-wise convolutions to construct end-to-end trainable shift-based modules, with a hyperparameter characterizing the tradeoff between accuracy and efficiency. To demonstrate the operation’s efficacy, we replace ResNet’s 3x3 convolutions with shift-based modules for improved CIFAR10 and CIFAR100 accuracy using 60% fewer parameters; we additionally demonstrate the operation’s resilience to parameter reduction on ImageNet, outperforming ResNet family members. We finally show the shift operation’s applicability across domains, achieving strong performance with fewer parameters on classification, face verification and style transfer.

## 1. Introduction and Related Work

Convolutional neural networks (CNNs) are ubiquitous in computer vision tasks, including image classification, object detection, face recognition, and style transfer. These tasks enable many emerging mobile applications and Internet-of-Things (IOT) devices; however, such devices have significant memory constraints and restrictions on the size of over-the-air updates (e.g. 100-150MB). This in turn imposes constraints on the size of the CNNs used in these applications. For this reason, we focus on reducing CNN model size while retaining accuracy, on applicable tasks.

CNNs rely on spatial convolutions with kernel sizes of 3x3 or larger to aggregate spatial information within an image. However, spatial convolutions are very expensive in both computation and model size, each of which grows quadratically with respect to kernel size. In the VGG-16

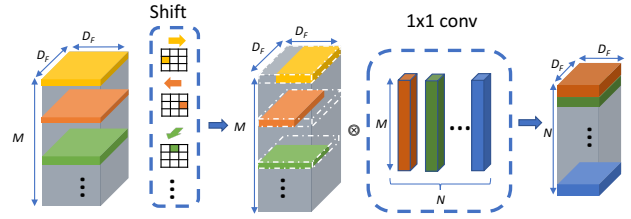


Figure 1: Illustration of a shift operation followed by a 1x1 convolution. The shift operation adjusts data spatially and the 1x1 convolution mixes information across channels.

model [19], 3x3 convolutions account for 15 million parameters, and the *fc1* layer, effectively a 7x7 convolution, accounts for 102 million parameters.

Several strategies have been adopted to reduce the size of spatial convolutions. ResNet[6] employs a “bottleneck module,” placing two 1x1 convolutions before and after a 3x3 convolution, reducing its number of input and output channels. Despite this, 3x3 convolutional layers still account for 50% of all parameters in ResNet models with bottleneck modules. SqueezeNet [9] adopts a “fire module,” where the outputs of a 3x3 convolution and a 1x1 convolution are concatenated along the channel dimension. Recent networks such as ResNext [26], MobileNet [7], and Xception [1] adopt group convolutions and depth-wise separable convolutions as alternatives to standard spatial convolutions. In theory, depth-wise convolutions require less computation. However, it is difficult to implement depth-wise convolutions efficiently in practice, as their arithmetic intensity (ratio of FLOPs to memory accesses) is too low to efficiently utilize hardware. Such a drawback is also mentioned in [29, 1]. ShuffleNet [29] integrates depth-wise convolutions, point-wise group convolutions, and channel-wise shuffling to further reduce parameters and complexity. In another work, [12] inherits the idea of a separable convolution to freeze spatial convolutions and learn only point-wise convolutions. This does reduce the number of learnable parameters but falls short of saving FLOPs or model size.

Our approach is to sidestep spatial convolutions entirely.

\* Authors contributed equally.

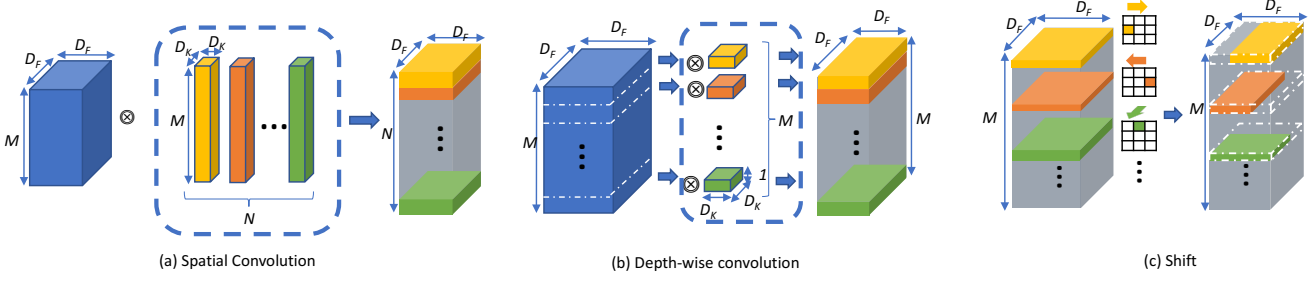


Figure 2: Illustration of (a) spatial convolutions, (b) depth-wise convolutions and (c) shift. In (c), the 3x3 grids denote a shift matrix with a kernel size of 3. The lighted cell denotes a 1 at that position and white cells denote 0s.

In this paper, we present the *shift operation* (Figure 1) as an alternative to spatial convolutions. The shift operation moves each channel of its input tensor in a different spatial direction. A *shift-based module* interleaves shift operations with point-wise convolutions, which further mixes spatial information across channels. Unlike spatial convolutions, the shift operation itself requires zero FLOPs and zero parameters. As opposed to depth-wise convolutions, shift operations can be easily and efficiently implemented.

Our approach is orthogonal to model compression [4], tensor factorization [27] and low-bit networks [16]. As a result, any of these techniques could be composed with our proposed method to further reduce model size.

We introduce a new hyperparameter for shift-based modules, “expansion”  $\mathcal{E}$ , corresponding to the tradeoff between FLOPs/parameters and accuracy. This allows practitioners to select a model according to specific device or application requirements. Using shift-based modules, we then propose a new family of architectures called *ShiftNet*. To demonstrate the efficacy of this new operation, we evaluate ShiftNet on several tasks: image classification, face verification, and style transfer. Using significantly fewer parameters, ShiftNet attains competitive performance.

## 2. The Shift Module and Network Design

We first review the standard spatial and depth-wise convolutions illustrated in Figure 2. Consider the spatial convolution in Figure 2(a), which takes a tensor  $F \in \mathbb{R}^{D_F \times D_F \times M}$  as input. Let  $D_F$  denote the height and width and  $M$  denote the channel size. The kernel of a spatial convolution is a tensor  $K \in \mathbb{R}^{D_K \times D_K \times M \times N}$ , where  $D_K$  denotes the kernel’s spatial height and width, and  $N$  is the number of filters. For simplicity, we assume the stride is 1 and that the input/output have identical spatial dimensions. Then, the spatial convolution outputs a tensor  $G \in \mathbb{R}^{D_F \times D_F \times N}$ , which can be computed as

$$G_{k,l,n} = \sum_{i,j,m} K_{i,j,m,n} F_{k+\hat{i},l+\hat{j},m}, \quad (1)$$

where  $\hat{i} = i - \lfloor D_F/2 \rfloor, \hat{j} = j - \lfloor D_F/2 \rfloor$  are the re-centered spatial indices;  $k, l$  and  $i, j$  index along spatial dimensions and  $n, m$  index into channels. The number of parameters required by a spatial convolution is  $M \times N \times D_K^2$  and the computational cost is  $M \times N \times D_K^2 \times D_F^2$ . As the kernel size  $D_K$  increases, we see the number of parameters and computational cost grow quadratically.

A popular variant of the spatial convolution is a depth-wise convolution [7, 1], which is usually followed by a point-wise convolution (1x1 convolution). Altogether, the module is called the depth-wise separable convolution. A depth-wise convolution, as shown in Figure 2(b), aggregates spatial information from a  $D_K \times D_K$  patch within each channel, and can be described as

$$\hat{G}_{k,l,m} = \sum_{i,j} \hat{K}_{i,j,m} F_{k+\hat{i},l+\hat{j},m}, \quad (2)$$

where  $\hat{K} \in \mathbb{R}^{D_F \times D_F \times M}$  is the depth-wise convolution kernel. This convolution comprises  $M \times D_K^2$  parameters and  $M \times D_K^2 \times D_F^2$  FLOPs. As in standard spatial convolutions, the number of parameters and computational cost grow quadratically with respect to the kernel size  $D_K$ . Finally, point-wise convolutions mix information across channels, giving us the following output tensor

$$G_{k,l,n} = \sum_m P_{m,n} \hat{G}_{k,l,m}, \quad (3)$$

where  $P \in \mathbb{R}^{M \times N}$  is the point-wise convolution kernel.

In theory, depth-wise convolution requires less computation and fewer parameters. In practice, however, this means memory access dominates computation, thereby limiting the use of parallel hardware. For standard convolutions, the ratio between computation vs. memory access is

$$\frac{M \times N \times D_F^2 \times D_K^2}{D_F^2 \times (M + N) + D_K^2 \times M \times N}, \quad (4)$$

while for depth-wise convolutions, the ratio is

$$\frac{M \times D_F^2 \times D_K^2}{D_F^2 \times 2M + D_K^2 \times M}. \quad (5)$$

A lower ratio here means that more time is spent on memory accesses, which are several orders of magnitude slower and more energy-consuming than FLOPs. This drawback implies an I/O-bound device will be unable to achieve maximum computational efficiency.

## 2.1. The Shift Operation

The shift operation, as illustrated in Figure 2(c), can be viewed as a special case of depth-wise convolutions. Specifically, it can be described logically as:

$$\tilde{G}_{k,l,m} = \sum_{i,j} \tilde{K}_{i,j,m} F_{k+\hat{i},l+\hat{j},m}. \quad (6)$$

The kernel of the shift operation is a tensor  $\tilde{K} \in \mathbb{R}^{D_F \times D_F \times M}$  such that

$$\tilde{K}_{k,l,m} = \begin{cases} 1, & \text{if } i = i_m \text{ and } j = j_m, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Here  $i_m, j_m$  are channel-dependent indices that assign one of the values in  $\tilde{K}_{:, :, m} \in \mathbb{R}^{D_K \times D_K}$  to be 1 and the rest to be 0. We call  $\tilde{K}_{:, :, m}$  a shift matrix.

For a shift operation with kernel size  $D_K$ , there exist  $D_K^2$  possible shift matrices, each of them corresponding to a shift direction. If the channel size  $M$  is no smaller than  $D_K^2$ , we can construct a shift matrix that allows each output position  $(k, l)$  access to all values within a  $D_K \times D_K$  window in the input. We can then apply another point-wise convolution per Eq. (3) to exchange information across channels.

Unlike spatial and depth-wise convolutions, the shift operation itself does not require parameters or floating point operations (FLOPs). Instead, it is a series of memory operations that adjusts channels of the input tensor in certain directions. A more sophisticated implementation can fuse the shift operation with the following 1x1 convolution, where the 1x1 convolution directly fetches data from the shifted address in cache. With such an implementation, we can aggregate spatial information using shift operations, for free.

## 2.2. Constructing Shift Kernels

For a given kernel size  $D_K$  and channel size  $M$ , there exists  $D_K^2$  possible shift directions, making  $(D_K^2)^M$  possible shift kernels. An exhaustive search over this state space for the optimal shift kernel is prohibitively expensive.

To reduce the state space, we use a simple heuristic: divide the  $M$  channels evenly into  $D_K^2$  groups, where each group of  $\lfloor M/D_K^2 \rfloor$  channels adopts one shift. We will refer to all channels with the same shift as a *shift group*. The remaining channels are assigned to the “center” group and are not shifted.

However, finding the optimal permutation, *i.e.*, how to map each channel- $m$  to a shift group, requires searching

a combinatorially large search space. To address this issue, we introduce a modification to the shift operation that makes input and output invariant to channel order: We denote a shift operation with channel permutation  $\pi$  as  $\mathcal{K}_\pi(\cdot)$ , so we can express Eq. (6) as  $\tilde{G} = \mathcal{K}_\pi(F)$ . We permute the input and output of the shift operation as

$$\tilde{G} = \mathcal{P}_{\pi_2}(\mathcal{K}_\pi(\mathcal{P}_{\pi_1}(F))) = (\mathcal{P}_{\pi_2} \circ \mathcal{K}_\pi \circ \mathcal{P}_{\pi_1})(F), \quad (8)$$

where  $\mathcal{P}_{\pi_i}$  are permutation operators and  $\circ$  denotes operator composition. However, permutation operators are discrete and therefore difficult to optimize. As a result, we process the input  $F$  to Eq. (8) by a point-wise convolution  $\mathcal{P}_1(F)$ . We repeat the process for the output  $\tilde{G}$  using  $\mathcal{P}_2(\tilde{G})$ . The final expression can be written as

$$\begin{aligned} G &= (\mathcal{P}_2 \circ \mathcal{P}_{\pi_2} \circ \mathcal{K}_\pi \circ \mathcal{P}_{\pi_1} \circ \mathcal{P}_1)(F) \\ &= ((\mathcal{P}_2 \circ \mathcal{P}_{\pi_2}) \circ \mathcal{K}_\pi \circ (\mathcal{P}_{\pi_1} \circ \mathcal{P}_1))(F) \\ &= (\hat{\mathcal{P}}_2 \circ \mathcal{K}_\pi \circ \hat{\mathcal{P}}_1)(F), \end{aligned} \quad (9)$$

where the final step holds, as there exists a bijection from the set of all  $\mathcal{P}_i$  to the set of all  $\hat{\mathcal{P}}_i$ , since the permutation operator  $\mathcal{P}_{\pi_i}$  is bijective by construction. As a result, it suffices to learn  $\hat{\mathcal{P}}_1$  and  $\hat{\mathcal{P}}_2$  directly. Therefore, this augmented shift operation Eq. (9) can be trained with stochastic gradient descent end-to-end, without regard for channel order. So long as the shift operation is sandwiched between two point-wise convolutions, different permutations of shifts are equivalent. Thus, we can choose an arbitrary permutation for the shift kernel, after fixing the number of channels for each shift direction.

## 2.3. Shift-based Modules

First, we define a *module* to be a collection of layers that perform a single function, *e.g.* ResNet’s bottleneck *module* or SqueezeNet’s fire *module*. Then, we define a *group* to be a collection of repeated modules.

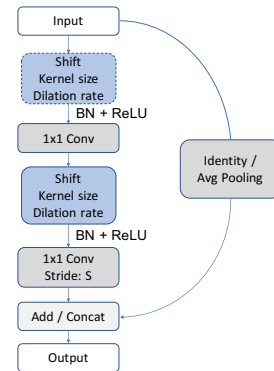


Figure 3: Illustration of the *Conv-Shift-Conv CSC* module and the *Shift-Conv-Shift-Conv (SC<sup>2</sup>)* module.

Based on the analysis in previous sections, we propose a module using shift operations as shown in Figure 3. The input tensor is first processed by point-wise convolutions. Then, we perform a shift operation to redistribute spatial information. Finally, we apply another set of point-wise convolutions to mix information across channels. Both sets of point-wise convolutions are followed by batch normalization and a non-linear activation function (ReLU). Following ShuffleNet [29], we use an additive residual connection when input and output are of the same shape, and use average pooling with concatenation when we down-sample the input spatially and double the output channels. We refer to this as a *Conv-Shift-Conv* or *CSC* module. A variant of this module includes another shift operation before the first point-wise convolution; we refer to this as the *Shift-Conv-Shift-Conv* or *SC<sup>2</sup>* module. This allows the designer to further increase the receptive field of the module.

As with spatial convolutions, shift modules are parameterized by several factors that control its behavior. We use the kernel size of the shift operation to control the receptive field of the *CSC* module. Akin to the dilated convolution, the “dilated shift” samples data at a spatial interval, which we define to be the *dilation rate*  $\mathcal{D}$ . The *stride* of the *CSC* module is defined to be the stride of the second point-wise convolution, so that spatial information is mixed in the shift operation before down-sampling. Similar to the bottleneck module used in ResNet, we use the “expansion rate”  $\mathcal{E}$  to control the intermediate tensor’s channel size. With bottleneck modules, 3x3 convolutions in the middle are expensive computationally, forcing small intermediate channel sizes. However, the shift operation allows kernel size  $D_F$  adjustments without affecting parameter size and FLOPs. As a consequence, we can employ a shift module to allow larger intermediate channel sizes, where sufficient information can be gathered from nearby positions.

### 3. Experiments

We first assess the shift module’s ability to replace convolutional layers, and then adjust hyperparameter  $\mathcal{E}$  to observe tradeoffs between model accuracy, model size, and computation. We then construct a range of shift-based networks and investigate their performance for a number of different applications.

#### 3.1. Operation Choice and Hyperparameters

Using ResNet, we juxtapose the use of 3x3 convolutional layers with the use of *CSC* modules, by replacing all of ResNet’s basic modules (two 3x3 convolutional layers) with *CSC*s to make “ShiftResNet”. For ResNet and ShiftResNet, we use two Tesla K80 GPUs with batch size 128 and a starting learning rate of 0.1, decaying by a factor of 10 after 32k and 48k iterations, as in [6]. In these experiments, we use the CIFAR10 version of ResNet: a convolutional layer

with 16 3x3 filters; 3 groups of basic modules with output channels 16, 32, 64; and a final fully-connected layer. A basic module contains two 3x3 convolutional layers followed by batchnorm and ReLU in parallel with a residual connection. With ShiftResNet, each group contains several *CSC* modules. We use three ResNet models: in ResNet20, each group contains 3 basic modules. For ResNet56, each contains 5, and for ResNet110, each contains 7. By toggling the hyperparameter  $\mathcal{E}$ , the number of filters in the *CSC* module’s first set of 1x1 convolutions, we can reduce the number of parameters in “ShiftResNet” by nearly 3 times without any loss in accuracy, as shown in Table 1. Table 5 summarizes CIFAR10 and CIFAR100 results across all  $\mathcal{E}$  and ResNet models.

We next compare different strategies for parameter reduction. We reduce ResNet’s parameters to match that of ShiftResNet for some  $\mathcal{E}$ , denoted ResNet- $\mathcal{E}$  and ShiftResNet- $\mathcal{E}$ , respectively. We use two separate approaches: 1) module-wise: decrease the number of filters in each module’s first 3x3 convolutional layer; 2) net-wise: decrease every module’s input and output channels by the same factor. As Table 2 shows, convolutional layers are less resilient to parameter reduction, with the shift module preserving accuracy 8% better than both reduced ResNet models of the same size, on CIFAR100. In Table 3, we likewise find improved resilience on ImageNet as ShiftResNet achieve better accuracy with millions fewer parameters.

Table 4 shows that ShiftResNet consistently outperforms ResNet, when both are constrained to use 1.5x fewer parameters. Table 5 then includes all results. Figure 4 shows the tradeoff between CIFAR100 accuracy and number of parameters for the hyperparameter  $\mathcal{E} \in \{1, 3, 6, 9\}$  across both {ResNet, ShiftResNet} models using varying numbers of layers  $\ell \in \{20, 56, 110\}$ . Figure 5 examines the same set of possible models and hyperparameters but between CIFAR100 accuracy and the number of FLOPs. Both figures show that ShiftResNet models provide superior trade-off between accuracy and parameters/FLOPs.

Table 1: PARAMETERS FOR SHIFT VS CONVOLUTION, WITH FIXED ACCURACY ON CIFAR-100

| Model           | Top1 Acc      | FLOPs       | Params     |
|-----------------|---------------|-------------|------------|
| ShiftResNet56-3 | <b>69.77%</b> | <b>0.3M</b> | <b>54M</b> |
| ResNet56        | 69.27%        | 0.87M       | 151M       |

#### 3.2. ShiftNet

Even though ImageNet classification is not our primary goal, to further investigate the effectiveness of the shift operation, we use the proposed *CSC* module shown in Figure 3 to design a class of efficient models called ShiftNet and present its classification performance on standard benchmarks to compare with state-of-the-art small models.

Table 2: REDUCTION RESILIENCE FOR SHIFT VS CONVOLUTION, WITH FIXED PARAMETERS

| Model            | CIFAR-100 Acc | FLOPs      | Params      |
|------------------|---------------|------------|-------------|
| ShiftResNet110-1 | <b>67.84%</b> | 36M        | <b>203K</b> |
| ResNet110-1      | 60.44%        | <b>32M</b> | 211K        |

Table 3: REDUCTION RESILIENCE FOR SHIFT VS CONVOLUTION ON IMAGENET

| Shift50<br>Top1 / Top5 Acc | ResNet50<br>Top1 / Top5 Acc | Parameters<br>Shift50 / ResNet50 |
|----------------------------|-----------------------------|----------------------------------|
| <b>75.6 / 92.8</b>         | 75.1 / 92.5                 | 22M / 26M                        |
| <b>73.7 / 91.8</b>         | 73.2 / 91.6                 | 11M / 13M                        |
| <b>70.6 / 89.9</b>         | 70.1 / 89.9                 | 6.0M / 6.9M                      |

Here, we abbreviate “ShiftResNet50” as “Shift50”.

Table 4: PERFORMANCE ACROSS RESNET MODELS, WITH 1.5 FEWER PARAMETERS

| No. Layers | ShiftResNet-6<br>CIFAR100 Top 1 | ResNet<br>CIFAR100 Top 1 |
|------------|---------------------------------|--------------------------|
| 20         | <b>68.64%</b>                   | 66.25%                   |
| 56         | <b>72.13%</b>                   | 69.27%                   |
| 110        | <b>72.56%</b>                   | 72.11%                   |

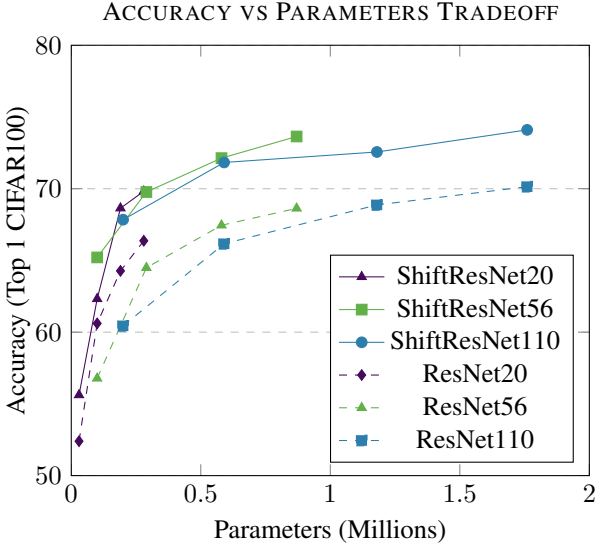


Figure 4: This figure shows that ShiftResNet family members are significantly more efficient than their corresponding ResNet family members. Tradeoff curves further to the top left are more efficient, with higher accuracy per parameter. For ResNet, we take the larger of two accuracies between module-wise and net-wise reduction results.

Since an external memory access consumes 1000x more energy than a single arithmetic operation [5], our primary goal in designing ShiftNet is to optimize the number of pa-

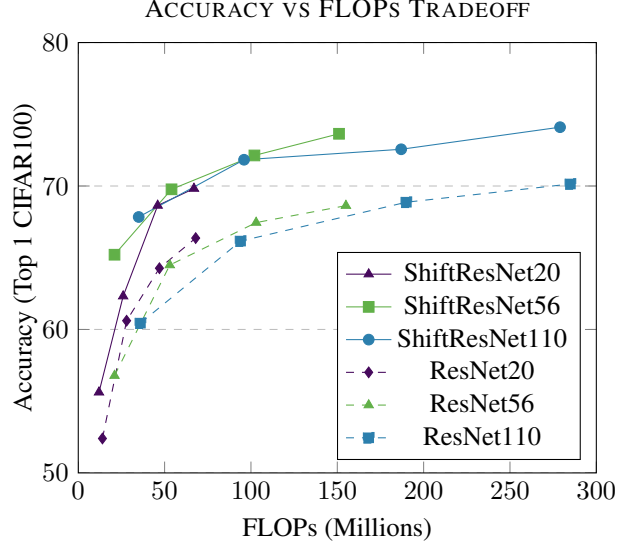


Figure 5: Tradeoff curves further to the top left are more efficient, with higher accuracy per FLOP. This figure shows ShiftResNet is more efficient than ResNet, in FLOPs.

rameters and thereby to reduce memory footprint. In addition to the general desirability of energy efficiency, the main targets of ShiftNet are mobile and IOT applications, where memory footprint, even more so than FLOPs, are a primary constraint. In these application domains small models can be packaged in mobile and IOT applications that are delivered within the 100-150MB limit for mobile over-the-air updates. In short, our design goal for ShiftNets is to attain competitive accuracy with fewer parameters.

The ShiftNet architecture is described in Table 6. Since parameter size does not grow with shift kernel size, we use a larger kernel size of 5 in earlier modules. We adjust the expansion parameter  $\mathcal{E}$  to scale the parameter size in each *CSC* module. We refer to the architecture described in Table 6 as ShiftNet-A. We shrink the number of channels in all *CSC* modules by 2 for ShiftNet-B. We then build a smaller and shallower network, with  $\{1, 4, 4, 3\}$  *CSC* modules in groups  $\{1, 2, 3, 4\}$  with channel sizes of  $\{32, 64, 128, 256\}$ ,  $\mathcal{E} = 1$  and kernel size is 3 for all modules. We name this shallow model ShiftNet-C. We train the three ShiftNet variants on the ImageNet 2012 classification dataset [17] with 1.28 million images and evaluate on the validation set of 50K images. We adopt data augmentations suggested by [1] and weight initializations suggested by [6]. We train our models for 90 epochs on 64 Intel KNL instances using Intel Caffe [10] with a batch size of 2048, an initial learning rate of 0.8, and learning rate decay by 10 every 30 epochs.

In Table 7, we show classification accuracy and number of parameters for ShiftNet and other state-of-the-art models. We compare ShiftNet- $\{A, B, C\}$  with 3 groups of models

Table 5: SHIFT OPERATION ANALYSIS USING CIFAR10 AND CIFAR100

| Model | $\mathcal{E}$ | ShiftResNet<br>CIFAR10 / 100 Accuracy | ResNet (Module)<br>CIFAR10 / 100 Accuracy | ResNet (Net)<br>CIFAR100 Accuracy | ShiftResNet<br>Parameters / FLOPs | Reduction Rate<br>Parameters / FLOPs |
|-------|---------------|---------------------------------------|---|-----------------------------------|-----------------------------------|--------------------------------------|
| 20    | 1             | 86.66% / 55.62%                       | 85.54% / 52.40%                           | 49.58%                            | 0.03 / 12                         | 7.8 / 5.5                            |
| 20    | 3             | 90.08% / 62.32%                       | 88.33% / 60.61%                           | 58.16%                            | 0.10 / 26                         | 2.9 / 2.6                            |
| 20    | 6             | 90.59% / 68.64%                       | 90.09% / 64.27%                           | 63.22%                            | 0.19 / 46                         | 1.5 / 1.4                            |
| 20    | 9             | <b>91.69% / 69.82%</b>                | 91.35% / 66.25%                           | 66.25%                            | 0.28 / 67                         | 0.98 / 1.0                           |
| 56    | 1             | 89.71% / 65.21%                       | 87.46% / 56.78%                           | 56.62%                            | 0.10 / 21                         | 8.4 / 7.0                            |
| 56    | 3             | 92.11% / 69.77%                       | 89.40% / 62.53%                           | 64.49%                            | 0.29 / 54                         | 2.9 / 2.8                            |
| 56    | 6             | 92.69% / 72.13%                       | 89.89% / 61.99%                           | 67.45%                            | 0.58 / 102                        | 1.5 / 1.5                            |
| 56    | 9             | <b>92.74% / 73.64%</b>                | 92.01% / 69.27%                           | 69.27%                            | 0.87 / 151                        | 0.98 / 1.0                           |
| 110   | 1             | 90.34% / 67.84%                       | 76.82% / 39.90%                           | 60.44%                            | 0.20 / 35                         | 8.5 / 7.8                            |
| 110   | 3             | 91.98% / 71.83%                       | 74.30% / 40.52%                           | 66.61%                            | 0.59 / 96                         | 2.9 / 2.9                            |
| 110   | 6             | 93.17% / 72.56%                       | 79.02% / 40.23%                           | 68.87%                            | 1.18 / 187                        | 1.5 / 1.5                            |
| 110   | 9             | <b>92.79% / 74.10%</b>                | 92.46% / 72.11%                           | 72.11%                            | 1.76 / 279                        | 0.98 / 1.0                           |

Note that the ResNet-9 results are replaced with accuracy of the original model. The number of parameters holds for both ResNet and ShiftResNet across CIFAR10, CIFAR100. FLOPs are computed for ShiftResNet. All accuracies are Top 1. “Reduction Rate” is the original ResNet’s parameters/flops over the new ShiftResNet’s parameters/flops.

Table 6: SHIFTNET ARCHITECTURE

| Group | Type/<br>Stride | Kernel | $\mathcal{E}$ | Output<br>Channel | Repeat |
|-------|-----------------|--------|---------------|-------------------|--------|
| -     | Conv / s2       | 7×7    | -             | 32                | 1      |
| 1     | CSC / s2        | 5×5    | 4             | 64                | 1      |
|       | CSC / s1        | 5×5    | 4             |                   | 4      |
| 2     | CSC / s2        | 5×5    | 4             | 128               | 1      |
|       | CSC / s1        | 5×5    | 3             |                   | 5      |
| 3     | CSC / s2        | 3×3    | 3             | 256               | 1      |
|       | CSC / s1        | 3×3    | 2             |                   | 6      |
| 4     | CSC / s2        | 3×3    | 2             | 512               | 1      |
|       | CSC / s1        | 3×3    | 1             |                   | 2      |
| -     | Avg Pool        | 7×7    | -             | 512               | 1      |
| -     | FC              | -      | -             | 1k                | 1      |

with similar levels of accuracy. In the first group, ShiftNet-A is **34X** smaller than VGG-16, while the top-1 accuracy drop is only 1.4%. ShiftNet-B’s top-1 accuracy is 2.5% worse than its MobileNet counterpart, but it uses fewer parameters. We compare ShiftNet-C with SqueezeNet and AlexNet, and we can achieve better accuracy with 2/3 the number of SqueezeNet’s parameters, and **77X** smaller than AlexNet.

### 3.3. Face Embedding

We continue to investigate the shift operation for different applications. Face verification and recognition are becoming increasingly popular on mobile devices. Both functionalities rely on face embedding, which aims to learn a mapping from face images to a compact embedding in Euclidean space, where face similarity can be directly measured by embedding distances. Once the space has been

Table 7: SHIFTNET RESULTS ON IMAGENET

| Model                   | Accuracy<br>Top-1 / Top-5 | Parameters<br>(Millions) |
|-------------------------|---------------------------|--------------------------|
| VGG-16 [19]             | <b>71.5 / 90.1</b>        | 138                      |
| GoogleNet [20]          | 69.8 / -                  | 6.8                      |
| ShiftResNet-0.25 (ours) | 70.6 / 89.9               | 6.0                      |
| ShuffleNet-2× [29]*     | 70.9 / -                  | 5.6                      |
| 1.0 MobileNet-224 [7]   | 70.6 / -                  | 4.2                      |
| Compact DNN [25]        | 68.9 / 89.0               | 4.1                      |
| ShiftNet-A (ours)       | 70.1 / 89.7               | <b>4.1</b>               |
| 0.5 MobileNet-224 [7]   | <b>63.7</b> / -           | 1.3                      |
| ShiftNet-B (ours)       | 61.2 / 83.6               | <b>1.1</b>               |
| AlexNet [13]            | 57.2 / 80.3               | 60                       |
| SqueezeNet [9]          | 57.5 / 80.3               | 1.2                      |
| ShiftNet-C (ours)       | <b>58.8 / 82.0</b>        | <b>0.78</b>              |

\* Estimated according to the model description in [29]

generated, various face-learning tasks, such as facial recognition and verification, can be easily accomplished by standard machine learning methods with feature embedding. Mobile devices have limited computation resources, therefore creating small neural networks for face embedding is a necessary step for mobile deployment.

FaceNet [18] is one state-of-the-art face embedding approach. The original FaceNet is based on Inception-Resnet-v1 [21], which contains 28.5 million parameters, making it difficult to be deployed on mobile devices. We propose a new model ShiftFaceNet based on ShiftNet-C from the previous section, which only contains 0.78 million parameters.

Following [15], we train FaceNet and ShiftFaceNet by combining the softmax loss with center loss [23]. We evaluate the proposed method on three datasets for face verifi-



Table 8: FACE VERIFICATION ACCURACY FOR SHIFT-FACENET VS FACENET [18].

|     | Accuracy $\pm$ STD (%) |                | Area under curve (%) |              |
|-----|------------------------|----------------|----------------------|--------------|
|     | FaceNet                | ShiftFaceNet   | FaceNet              | ShiftFaceNet |
| LFW | 97.1 $\pm$ 1.3         | 96.0 $\pm$ 1.4 | 99.5                 | 99.4         |
| YTF | 92.0 $\pm$ 1.1         | 90.1 $\pm$ 0.9 | 97.3                 | 96.1         |
| MSC | 79.2 $\pm$ 1.7         | 77.6 $\pm$ 1.7 | 85.6                 | 84.4         |

Table 9: FACE VERIFICATION PARAMETERS FOR SHIFT-FACENET VS FACENET [18].

| Model             | FaceNet | ShiftFaceNet |
|-------------------|---------|--------------|
| Params (Millions) | 28.5    | 0.78         |

cation: given a pair of face images, a distance threshold is selected to classify the two images belonging to the same or different entities. The LFW dataset [8] consists of 13,323 web photos with 6,000 face pairs. The YTF dataset [24] includes 3,425 with 5,000 video pairs for video-level face verification. The MS-Celeb-1M dataset (MSC) [3] comprises 8,456,240 images for 99,892 entities. In our experiments, we randomly select 10,000 entities from MSC as our training set, to learn the embedding space. We test on 6,000 pairs from LFW, 5,000 pairs from YTF and 100,000 pairs randomly generated from MSC, excluding the training set. In the pre-processing step, we detect and align all faces using a multi-task CNN [28]. Following [18], the similarity between two videos is computed as the average similarity of 100 random pairs of frames, one from each video. Results are shown in Table 8. Parameter size for the original FaceNet and our proposed ShiftFaceNet is shown in Table 9. With ShiftFaceNet, we are able to reduce the parameter size by **35X**, with at most 2% drop of accuracy in above three verification benchmarks.

### 3.4. Style Transfer

Artistic style transfer is another popular application on mobile devices. It is an image transformation task where the goal is to combine the *content* of one image with the *style* of another. Although this is an ill-posed problem without definite quantitative metrics, a successful style transfer requires that networks capture both minute textures and holistic semantics, for content and style images.

Following [2, 11], we use perceptual loss functions to train a style transformer. In our experiment, we use a VGG-16 network pretrained on ImageNet to generate the perceptual loss. The original network trained by Johnson *et al.* [11] consists of three downsampling convolutional layers, five residual modules, and three upsampling convolutional layers. All non-residual convolutions are followed by an instance normalization layer [22]. In our experiments, we replace all but the first and last convolution layers with shifts followed by 1x1 convolutions. We train the network on the COCO [14] dataset using the previously reported hyperpa-

Table 10: STYLE TRANSFER: SHIFT VS CONVOLUTION

| Model             | Original | Shift |
|-------------------|----------|-------|
| Params (Millions) | 1.9      | 0.3   |

rameter settings  $\lambda_s \in \{1e10, 5e10, 1e11\}$ ,  $\lambda_c = 1e5$ . By replacing convolutions with shifts, we achieve an overall **6X** reduction in the number of parameters with minimal degradation in image quality. Examples of stylized images generated by original and shift based transformer networks can be found in Figure 6.

## 4. Discussion

In our experiments, we demonstrate the shift operation’s effectiveness as an alternative to spatial convolutions. Our construction of shift groups is trivial: we assign a fixed number of channels to each shift. However, this assignment is largely uninformed. In this section, we explore more informed allocations and potential improvements.

An ideal channel allocation should at least have the following 2 properties: 1) Features in the same shift group should not be redundant. We can measure redundancy by checking the correlation between channel activations within a shift group. 2) Each shifted feature should have a non-trivial contribution to the output. We can measure the contribution of channel- $m$  to the output by  $\|P_{m,:}\|_2$ , *i.e.*, the  $l_2$  norm of the  $m$ -th row of the second point-wise convolution kernel in a *CSC* module.

### 4.1. Channel Correlation Within Shifts

We analyze one *CSC* module with 16 input/output channels and an expansion of 9, from a trained ShiftResnet20 model. We first group channels by shift. Say each shift group contains  $c$  channels. We consider activations of the  $c$  channels as a random variable  $X \in \mathbb{R}^c$ . We feed validation images from CIFAR100 into the network and record intermediate activations in the *CSC* module, to estimate correlation matrices  $\Sigma_{XX} \in \mathbb{R}^{c \times c}$ , a few of which are shown in figure 8. We use correlation between channels as a proxy for measuring redundancy. For example, if the correlation between two channels is above a threshold, we can remove one channel. In Figure 8, candidate pairs could stem from the bright green pixels slightly off-center in the first matrix.

### 4.2. Normalized Channel Contributions

We analyze the same *CSC* module from the section above. Consider the  $m$ -th channel’s contribution to the output of the module, we take the norm of the  $m$ -th row from the point-wise convolution kernel as  $\|P_{m,:}\|_2$  for an approximation of “contribution”. We compute the contribution of all 144 channels in the given *CSC* module, normalize their contribution and plot it in Figure 7. Note that the shift contributions are anisotropic, and the largest contributions fall into a cross, where horizontal information is accentuated.

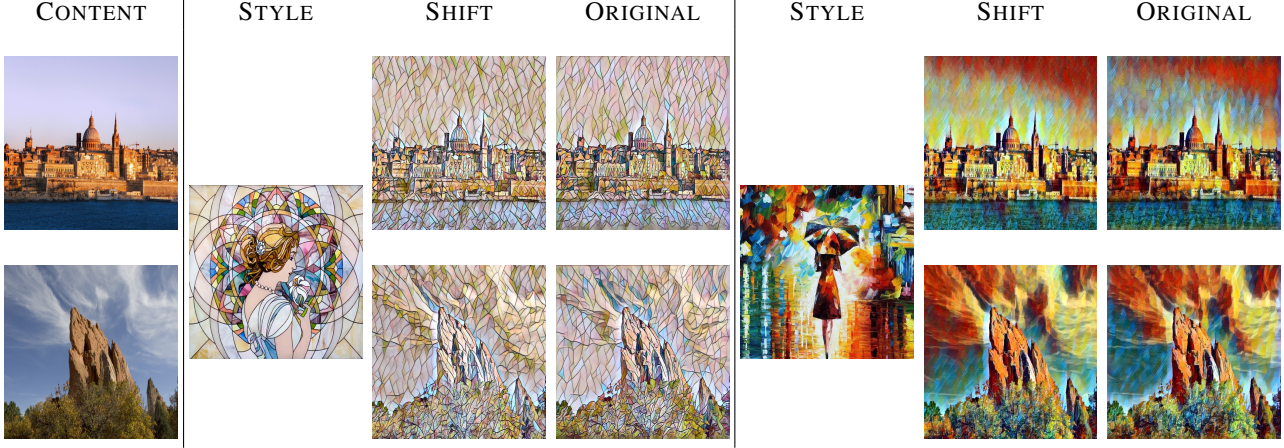


Figure 6: STYLE TRANSFER RESULTS USING SHIFNet

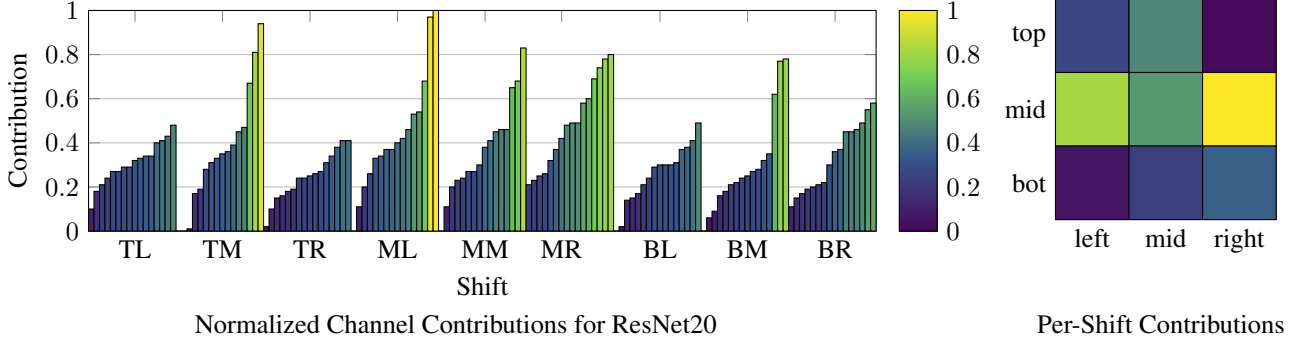


Figure 7: Left: We plot normalized channel contributions. Along the horizontal axis, we rank the channels in 9 groups of 16 (one group per shift) B=bottom, M=mid, T=top, R=right, L=left. Right: For each shift pattern, we plot the sum of its contributions normalized to the same scale. Both figures share the same color map, where yellow has the highest magnitude.

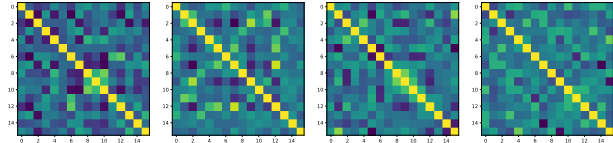


Figure 8: Example Channel Correlation Matrices Within each Shift Group

This suggests that better heuristics for allocating channels among shift groups may yield neural networks with even higher per-FLOP and per-parameter accuracy.

## 5. Conclusion

We present the shift operation, a zero-FLOP, zero-parameter, easy-to-implement alternative to convolutions for aggregation of spatial information. To start, we construct end-to-end-trainable modules using shift operations and pointwise convolutions. To demonstrate their efficacy and robustness, we replace ResNet’s convolutions with shift

modules for varying model size constraints, which increases accuracy by up to 8% with the same number of parameters/FLOPs and recovers accuracy with a third of the parameters/FLOPs. We then construct a family of shift-based neural networks. Among neural networks with around 4 million parameters, we attain competitive performance on a number of tasks, namely classification, face verification, and style transfer. In the future, we plan to apply ShiftNets to tasks demanding large receptive fields that are prohibitively expensive for convolutions, such as 4K image segmentation.

## Acknowledgement

This work was partially supported by the DARPA PERFECT program, Award HR0011-12-2-0016, together with ASPIRE Lab sponsor Intel, as well as lab affiliates HP, Huawei, Nvidia, and SK Hynix. This work has also been partially sponsored by individual gifts from BMW, Intel, and the Samsung Global Research Organization. We thank Fisher Yu, Xin Wang for valuable discussions. We thank Kostadin Ilov for providing system assistance.



## References

- [1] F. Chollet. Xception: Deep learning with depthwise separable convolutions. *arXiv preprint arXiv:1610.02357*, 2016.
- [2] L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015.
- [3] Y. Guo, L. Zhang, Y. Hu, X. He, and J. Gao. Ms-celeb-1m: Challenge of recognizing one million celebrities in the real world. *Electronic Imaging*, 2016(11):1–6, 2016.
- [4] S. Han, H. Mao, and W. J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.
- [5] S. Han, J. Pool, J. Tran, and W. J. Dally. Learning both weights and connections for efficient neural networks. *CoRR*, abs/1506.02626, 2015.
- [6] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.
- [7] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.
- [8] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Rethinking the inception architecture for computer vision. In *ECCV Workshops*, 2016.
- [9] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.
- [10] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 675–678. ACM, 2014.
- [11] J. Johnson, A. Alahi, and F. Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.
- [12] F. Juefei-Xu, V. N. Boddeti, and M. Savvides. Local binary convolutional neural networks. *arXiv preprint arXiv:1608.06049*, 2016.
- [13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
- [14] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [15] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *BMVC*, volume 1, page 6, 2015.
- [16] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pages 525–542. Springer, 2016.
- [17] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.
- [18] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823.
- [19] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.
- [22] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.
- [23] Y. Wen, K. Zhang, Z. Li, and Y. Qiao. A discriminative feature learning approach for deep face recognition. In *ECCV*, pages 499–515, 2016.
- [24] L. Wolf, T. Hassner, and I. Maoz. Face recognition in unconstrained videos with matched background similarity. In *CVPR*, pages 529–534, 2011.
- [25] C. Wu, W. Wen, T. Afzal, Y. Zhang, Y. Chen, and H. Li. A compact DNN: approaching googlenet-level accuracy of classification and domain adaptation. *CoRR*, abs/1703.04071, 2017.
- [26] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. *arXiv preprint arXiv:1611.05431*, 2016.
- [27] X. Yu, T. Liu, X. Wang, and D. Tao. On compressing deep models by low rank and sparse decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7370–7379, 2017.
- [28] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.
- [29] X. Zhang, X. Zhou, M. Lin, and J. Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. *arXiv preprint arXiv:1707.01083*, 2017.