Tied Block Convolution: Leaner and Better CNNs with Shared Thinner Filters

Xudong Wang and **Stella X. Yu**
UC Berkeley / ICSI
{xdwang, stellayu}@berkeley.edu

Abstract

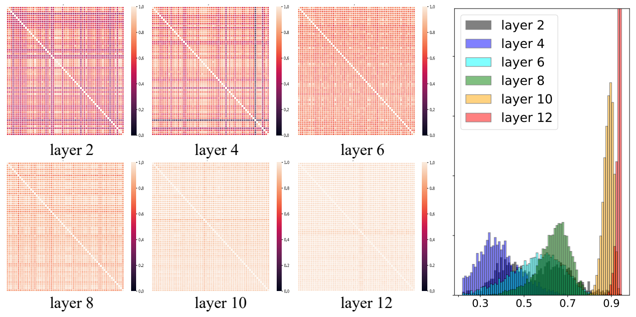
Convolution is the main building block of a convolutional neural network (CNN). We observe that an optimized CNN often has highly correlated filters as the number of channels increases with depth, reducing the expressive power of feature representations. We propose *Tied Block Convolution* (TBC) that shares the same thinner filter over equal blocks of channels and produces multiple responses with a single filter. The concept of TBC can also be extended to group convolution and fully connected layers, and can be applied to various backbone networks and attention modules.

Our extensive experimentation on classification, detection, instance segmentation, and attention demonstrates that TBC is consistently leaner and significantly better than standard convolution and group convolution. On attention, with 64× fewer parameters, our TiedSE performs on par with the standard SE. On detection and segmentation, TBC can effectively handle highly overlapping instances, whereas standard CNNs often fail to accurately aggregate information in the presence of occlusion and result in multiple redundant partial object proposals. By sharing filters across channels, TBC reduces correlation and delivers a sizable gain of 6% in the average precision for object detection on MS-COCO when the occlusion ratio is 80%. Our code is publicly available.

Introduction

Convolution is the main building block of a convolutional neural network (CNN), which has been widely successful on image classification (Krizhevsky, Sutskever, and Hinton 2012; He et al. 2016; Xie et al. 2017; Simonyan and Zisserman 2014), object detection (Girshick 2015; Ren et al. 2015; He et al. 2017), image segmentation (Kirillov et al. 2019; Long, Shelhamer, and Darrell 2015; Chen et al. 2017, 2018) and action recognition (Ji et al. 2012; Wang et al. 2016; Carreira and Zisserman 2017; Wang et al. 2018). However, standard convolution is still costly in terms of computation, storage, and memory access. More importantly, an optimized CNN often develops highly correlated filters.

We can evaluate pairwise filter similarity in standard convolution (SC), using the cosine similarity of guided back-propagation patterns (Springenberg et al. 2014) averaged over a set of ImageNet images. Fig. 1 shows that the filter correlation increases with the layer depth: Filters at the same layer become more similar from early to later layers, reducing the expressive power of feature representations.

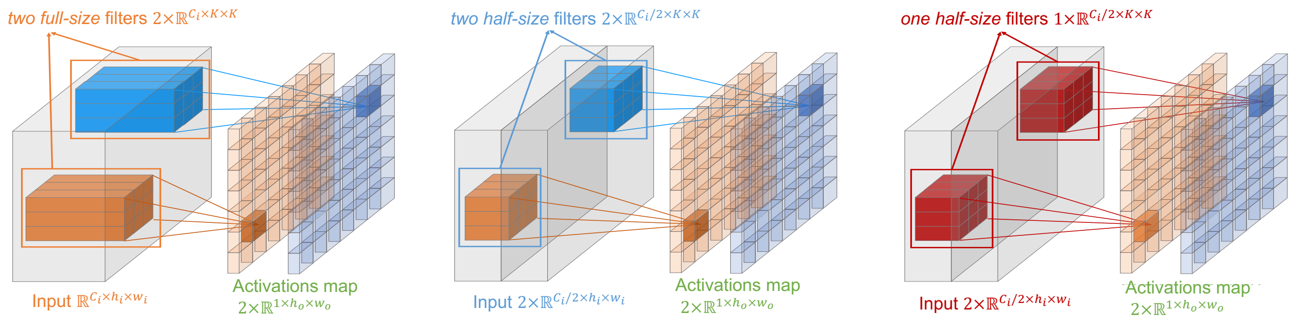


(a) Correlation matrix (b) Histograms

Figure 1: **Filters of an optimized CNN become more similar at an increasing depth**. (a) Correlation matrix of 64 randomly selected filters at selected layers of the VGG16 ImageNet classifier. We compute the similarity between two filters based on their guided back-propagation patterns (Springenberg et al. 2014) averaged on a set of images. As the layer goes deeper from 2 to 12, the pairwise similarity increases overall and the matrix turns from red to light yellow. (b) Normalized histograms of these pairwise filter similarities. As the number of channels increases with depth from 64 to 128 to 256, the curve shifts right and becomes far narrower, i.e., more filters become similar.

How to optimize a CNN with less redundancy has been studied (Howard et al. 2017; Zhang et al. 2018; Ma et al. 2018; Xie et al. 2017), often by exploring dependencies across space and channel dimensions. In SC, while each filter can have a reduced size spatially, it extends to the entire set of channels, whereas in group convolution (GC) (Krizhevsky, Sutskever, and Hinton 2012), a filter only convolves with a subset of input channels. Therefore, if there are B groups of input channels, each GC layer reduces the number of parameters B times by reducing the size of each filter by B times. Depth-wise convolution (DW) is an extreme case of GC, where each group only contains one channel, maximally reducing the parameter count.

While GC and DW are effective at reducing the model size, there is no correlation between filters, resulting in an isolated representation without cross-channel connections. Instead of simply reducing the size of each filter as in GC


two full-size filters $2\times\mathbb{R}^{C_i\times K\times K}$
Input $\mathbb{R}^{C_i\times h_i\times w_i}$
Activations map $2\times\mathbb{R}^{1\times h_o\times w_o}$
two half-size filters $2\times\mathbb{R}^{C_i/2\times K\times K}$
Input $2\times\mathbb{R}^{C_i/2\times h_i\times w_i}$
Activations map $2\times\mathbb{R}^{1\times h_o\times w_o}$
one half-size filters $1\times\mathbb{R}^{C_i/2\times K\times K}$
Input $2\times\mathbb{R}^{C_i/2\times h_i\times w_i}$
Activations map $2\times\mathbb{R}^{1\times h_o\times w_o}$

(a) Standard Convolution (b) Group Convolution (c) Tied Block Convolution

Figure 2: **Convolution operators.** To generate two activation maps, standard convolution requires *two full-size* filters and group convolution requires *two half-size* filters, however, our tied block convolution only requires *one half-size* filter, i.e., the number of parameters is reduced by 4×. The idea of TBC can also be applied to fully connected and group convolutional layers.

and DW, we further reduce redundancy by exploring the connections among filters on subsets of channels and consequently increasing the power of each filter.

Directly reducing the number of filters is known to reduce the model capacity (He et al. 2016). However, since SC filters become more similar (Fig. 1), we can reduce the *effective number* of filters by reusing them across channels.

We propose such a simple alternative called *tied block convolution* (TBC): We split C input channels into B equal blocks, and use a single block filter defined only on $\frac{C}{B}$ channels to produce B responses. While SC produces two responses with two full-size filters each spanning entire C channels, TBC at $B=2$ produces two responses with a single half-size filter spanning only $\frac{C}{2}$ channels (Fig. 2). TBC is GC shared across groups, and TBC at $B=1$ is SC.

Extending the concept of TBC in a straightforward fashion to the fully connected layer and the group convolution layer, we obtain tied block fully connected layer (TFC) and the tied block group convolution (TGC) respectively.

Our TBC utilizes each filter, memory access, and samples more effectively. **1)** At $B=2$, TBC obtains twice responses with one half-size thin filter, achieving 4 times model reduction. **2)** As the same thin filter is applied to each of the B blocks, TBC has more efficient memory access by utilizing GPU's parallel processing. **3)** Since each thin filter is trained on B times more samples, learning also becomes more effective. **4)** Since each set of TBC filters are applied to all input channels, TBC could aggregate global information across channels and model cross-channel dependencies.

While TBC is appealing in theory, its advantage over SC or GC in practice could depend upon neural network architectures. We apply TBC/TFC/TGC to various backbone networks, including ResNet (He et al. 2016), ResNeXt (Xie et al. 2017), SENet (Hu, Shen, and Sun 2018) and ResNeSt (Zhang et al. 2020), and propose their tied versions: *TiedResNet, TiedResNeXt, TiedSENet* and *TiedResNeSt*.

We conduct extensive experimentation on classification, detection, segmentation, and attention, demonstrating TBC/TGC/TFC's significant across-the-board performance gain over standard convolution, group convolution, and fully connected layer function. For example, TiedResNet consistently outperforms ResNet, ResNeXt and HRNetV2 (Wang et al. 2019) by a larger margin with a much leaner model (Fig. 6). We obtain similar performance boost and model reduction on a varity of frameworks, tasks and datasets.

Our empirical insight is that filter redundancy in an optimized CNN not only reduces the effective model capacity, but also makes it unable to capture diverse outputs and thereby loses performance. For object detection on MS-COCO, standard CNNs often fail to accurately locate target object regions and aggregate useful information in the foreground. Consequently, there are multiple overlapping partial object proposals, preventing a single full object proposal to emerge from the proposal pool. Our TiedResNet can handle highly overlapping instances much better and increase the average precision (AP) by 6% (in particular, 8.3% in AP at IoU$=0.75$) when the occlusion ratio is 80%.

Related works

Backbone Networks. AlexNet (Krizhevsky, Sutskever, and Hinton 2012) is the first CNN success with significant accuracy gain on the ILSVRC competition. However, large kernels and fully connected layers greatly increase the model size. With smaller kernels, GoogleNet (Szegedy et al. 2015) and VGGNet (Simonyan and Zisserman 2014) only need 12 times fewer parameters to outperform (Krizhevsky, Sutskever, and Hinton 2012; Zeiler and Fergus 2014). However, large network depths cause vanishing gradient problems, later to be solved by the residual connection design in ResNet (He et al. 2016). Since the depth of a CNN model is no longer an issue, researchers have begun to explore how to use parameters more efficiently. At a comparable model complexity, ResNeXt (Xie et al. 2017) outperformes ResNet on many major tasks, mainly due to the use of efficient group convolution. With a careful design of the architecture, HRNetV2 (Wang et al. 2019) achieves the state-of-the-art performance on multiple major tasks. Compared to these works using either GC or SC, our TBC further utilizes the full potential of each thinner filter. We provide detailed comparisons with these networks.

Group-wise Convolution. Group convolution (GC) (Krizhevsky, Sutskever, and Hinton 2012) is proposed to remove filter redundancy. Since each GC filter only convolves with features in its group, with the same number of channels, this mechanism can reduce the number of parameters within each layer by a factor of B, where B is the number of groups. When the number of groups is the same as the number of input feature channels, GC becomes identical to depth-wise convolution (DW) (Howard et al. 2017). Both GC and DW greatly reduce the model redundancy by reducing the size of each filter. However, they do not exploit the correlation between (learned) filters.

As each filter in GC or DW only responds to a partial input feature map, the ability to integrate information across channel dimensions is reduced in GC and completely lost in DW. In contrast, our TBC filter is shared across all input channels, and thus its responses over subsets of channels become *comparable* and relatable. This mechanism also introduces another benefit: With only one fragmentation, TBC can take full advantage of the powerful parallel computing capabilities of GPUs.

Attention Modules. (Hu, Shen, and Sun 2018) introduces the squeeze-and-excitation (SE) module to adaptatively recalibrate channel-wise feature responses. (Cao et al. 2019) unifies SE and a non-local (Wang et al. 2018) module into a global context block (GCB). While SE and GCB are relatively light, SE (GCB) still counts for 10% (25%) of the model size. Our tied block convolution and tied fully connected layers can be integrated into various attention modules and significantly reduce the number of parameters: 2.53M vs 0.04M for SE and 10M vs 2.5M for GCB.

Tied Block Convolution Network Design

We first analyze TBC and TGC to guide us in network design. We also develop TFC and apply to attention modules.

TBC Formulation

Let $X \in \mathbb{R}^{c_i \times h_i \times w_i}$ and $\tilde{X} \in \mathbb{R}^{c_o \times h_o \times w_o}$ denote the input and output features respectively, where c, h, w are the number of channels, the height and width of feature maps respectively. The kernel size is $k \times k$ and the bias term is ignored for clarity.

Standard Convolution, denoted by $*$, can be formulated as:

$$\tilde{X} = X * W \tag{1}$$

where $W \in \mathbb{R}^{c_o \times c_i \times k \times k}$ is the SC kernel. The number of parameters for SC is thus: $c_o \times c_i \times k \times k$.

Group Convolution first divides input feature X into G equal-sized groups $X_1, ..., X_G$ with size $c_i/G \times h_i \times w_i$ per group. Each group shares the same convolutional filters W_g. The output of GC is computed as:

$$\tilde{X} = X_1 * W_1 \oplus X_2 * W_2 \oplus \cdots \oplus X_G * W_G \tag{2}$$

where $\oplus$ is the concatenation operation along the channel dimension, W_g is the convolution filters for group g, where $g \in \{1, \ldots, G\}$, $W_g \in \mathbb{R}^{\frac{c_o}{G} \times \frac{c_i}{G} \times k \times k}$. The number of parameters for GC is: $G \times \frac{c_o}{G} \times \frac{c_i}{G} \times k \times k$.

Tied Block Convolution reduces the *effective number* of filters by reusing filters across different feature groups with the following formula:

$$\tilde{X} = X_1 * W' \oplus X_2 * W' \oplus \cdots \oplus X_B * W' \tag{3}$$

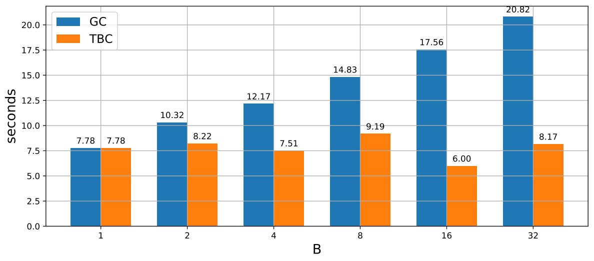


Figure 3: **TBC has a flat (vs. GC's linear) compute with respect to the block number**. The time cost of processing 1k iterations of each feature map using the RTX 2080Ti GPU is plotted against B. When B increases, GC increases the time cost almost linearly. In contrast, when using a larger B, TBC keeps a similar time cost. Different block numbers B were tested for GC and TBC, the total FLOPs at these values were fixed by changing the total filter number. When $B = 1$, GC and TBC are equal to SC. Input feature map size is 56×56×2048.

where $W' \in \mathbb{R}^{\frac{c_o}{B} \times \frac{c_i}{B} \times k \times k}$ is the TBC filters shared among all the groups. The parameter number is: $\frac{c_o}{B} \times \frac{c_i}{B} \times k \times k$.

TBC vs. GC. While TBC is GC with filters shared across groups, it has several major distinctions from GC in practical consequences (assume that the block number B is the same as the group number G).

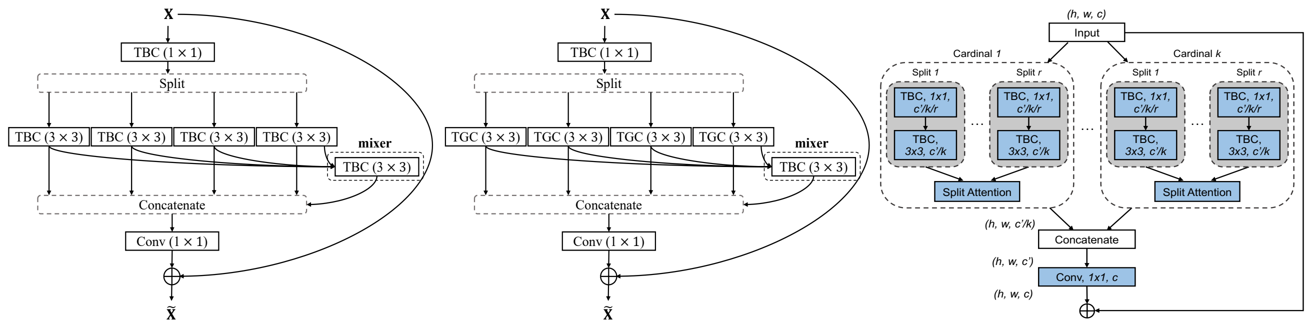
1. TBC has $B\times$ fewer parameters than GC.
2. TBC only has one fragmentation on GPU utilization, whereas GC has G fragmentations, greatly reducing the degree of parallelism. Fig.3 shows that the processing time increases linearly with the number of groups in GC, whereas our TBC keeps almost the same processing time.
3. TBC can better model cross-channel dependencies. Since each set of GC filters are only convolved on subsets of channels, GC has trouble comparing and aggregating information across channels. However, each set of TBC filters are applied to all input channels and can better model cross-channel dependencies.
4. TBC-based TiedResNet greatly surpasses GC-integrated ResNeXt on object detection and instance segmentation tasks. TiedResNet-S can even outperform ResNeXt with $2\times$ model size reduction, demonstrating that TiedResNet makes more effective use of model parameters.

Tied Block Group Convolution (TGC) The idea of tied block filtering can also be directly applied to group convolution, formulated as:

$$\begin{aligned}\tilde{X} =&(X_{11} * W'_1 \oplus \cdots \oplus X_{1B} * W'_1) \oplus \cdots \oplus \\ &(X_{G1} * W'_G \oplus \cdots \oplus X_{GB} * W'_G)\end{aligned} \tag{4}$$

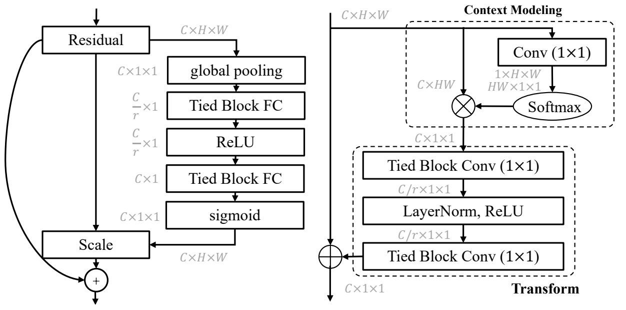
where $W'_g \in \mathbb{R}^{\frac{c_o}{BG} \times \frac{c_i}{BG} \times k \times k}$, $X_{gb} \in \mathbb{R}^{\frac{c_i}{BG} \times h_i \times w_i}$ is the divided feature map, $g \in [1, G]$ and $b \in [1, B]$.

Tied Block Fully Connected Layer (TFC) Convolution is a special case of fully connected (FC) layer, just as FC is a special case of convolution. We apply the same tied block filtering idea to FC. Tied block fully connected layer (TFC) shares the FC connections between equal blocks of input



(a) TiedResNet (b) TiedResNeXt (c) TiedResNeSt

Figure 4: **Diagram of bottleneck modules** for (a) TiedResNet with 4 splits (b) TiedResNeXt with 4 splits and (c) TiedResNeSt. Each tied block convolution (TBC) and tied block group convolution (TGC) has a specific block number.



(a) TiedSE (b) TiedGCB

Figure 5: **Diagram of Tied attention modules.** (a) TiedSE module replaces FC in the original squeeze-and-excitation (SE) module (Hu, Shen, and Sun 2018) to be TFC. (b) TiedGCB module replaces standard convolution in global context block (GCB) (Cao et al. 2019) with TBC.

channels. Like TBC, TFC could reduce B^2 times parameters and B times computational cost.

TBC/TGC in Bottleneck Modules

The ResNet/ResNeXt/ResNeSt bottleneck modules have 1×1 and 3×3 convolutional filters. We apply TBC/TGC differently as in Fig.4. For 3×3 in ResNet and ResNeXt, we split all the filters into groups; each group has its own TBC/TGC setting. This choice allows different levels of sharing and is motivated by network visualization works (Zeiler and Fergus 2014; Bau et al. 2017): Filters assume different roles at different layers and some are unique concept detectors (Agrawal, Carreira, and Malik 2015; Bau et al. 2017). For the 1×1 convolutions at the entry and the exit of bottlenecks, we replace the entry one by TBC with $B=2$ to allow filter sharing, while maintaining the exit convolution to aggregate information across channels. Since ResNeSt replaces 3×3 convolutions to be multi-path and split attention modules with k cardinals, 3×3 convolutions occupy less proportion of the overall model complexity. Therefore, we only replace all 3×3 convolution to be TBC with $B=2$ as in 1×1 convolution. Further increase of B would only marginally reduce the model size but greatly reduce the model performance.

The default setting for TiedResNet-50 (TiedResNeXt-50) is 4 splits with base width of 32 (64), i.e. 4s×32w (4s×64w), and the default setting for TiedResNet-S (TiedResNeXt-50-S) is 4s×18w (4s×36w). Our TiedBottleNeck reaches more than 1% performance improvement in term of top-1 accuracy on ImageNet-1K. However, losing cross-channel integration could weaken the model. To add it back, we introduce a mixer that fuses outputs of multiple splits. Introducing the mixer increases performance by another 0.5%. The input to the mixer can be either concatenation or element-wise sum of split outputs. Table 6 shows that element-wise sum has a better trade-off.

TBC and TFC in Attention Modules

We apply TBC and TFC to attention modules such as SE (Hu, Shen, and Sun 2018) and GCB (Cao et al. 2019), by simply replacing SC and FC with their tied block counterparts (Fig. 5). Both designs significantly reduce the number of parameters without dropping performance.

Experimental Results

We conduct extensive tests of TBC, TGC and TFC on major benchmarks for object recognition, object detection, instance segmentation and attention.

ImageNet Classification

Implementation. We follow standard practices and perform data augmentation with random cropping to size 224×224 pixels (He et al. 2016). We train the network using SGD with a momentum of 0.9 and a mini-batch of 256 on 8 GPUs. The learning rate is initially set to 0.1 and then decayed 10× every 30 epochs for a total of 100 epochs.

Performance gain. Table 1 compares the recognition accuracy of multiple models on ImageNet-1k (Deng et al. 2009) validation set. In Table 1, TiedResNet50-S beats ResNet50 in terms of top-1 accuracy with only 60% flops and 54% parameters, likewise for TiedResNet101-S. With similar model complexity, TiedResNet50 and TiedResNet101 can beat benchmarks by more than 1.5% and 1.4% separately with 10% parameter reduction. Similar tendency can be observed for TiedResNeXt and TiedSENet. To further prove the effectiveness of TBC, we integrate it with current SOTA model ResNeSt. With only 59% of parameters and 82% of computation cost, TiedResNeSt-50-S obtains better performance than ResNeSt-50-S on ImageNet-1k.

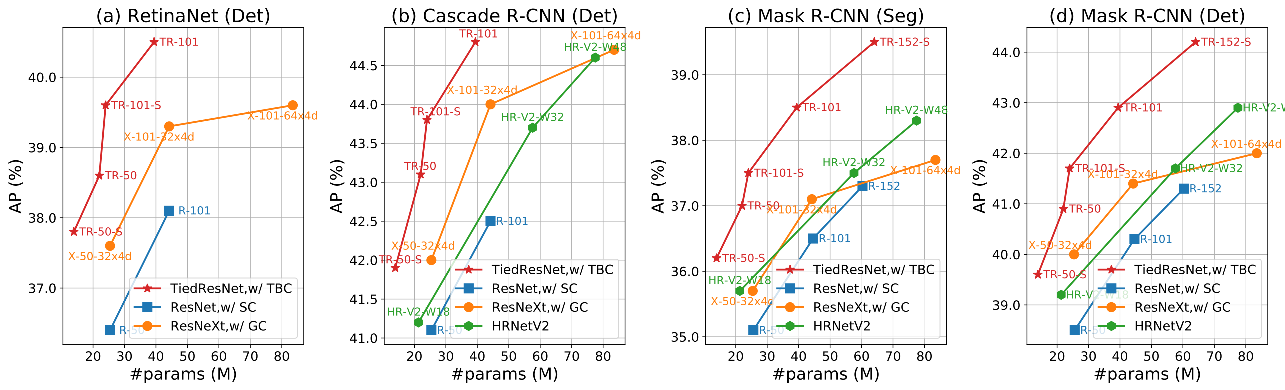


Figure 6: TiedResNet consistently outperforms ResNet, ResNeXt and HRNetV2 with much fewer parameters, experimented on *single-stage* detector RetinaNet and *two-stage* detectors Cascade R-CNN and Mask R-CNN. We plot #params of backbones vs. their Average Precision on **object detection and instance segmentation tasks of MS-COCO *val-2017***.

model	params(M)	GFlops	top-1(%)	top-5(%)
ResNet50 (He et al. 2016)				
baseline	25.6	4.2	76.2	92.9
TiedResNet50-S	**13.9** **(54%)**	**2.5** **(60%)**	76.2	92.9
TiedResNet50	22.0 (86%)	4.4 (105%)	**77.6**	**93.6**
ResNet101 (He et al. 2016)				
baseline	44.6	7.9	77.4	93.6
TiedResNet101-S	**24.0** **(54%)**	**4.8** **(61%)**	77.7	93.8
TiedResNet101	39.4 (88%)	8.6 (109%)	**78.8**	**94.2**
ResNeXt101-32×8d (Xie et al. 2017)				
baseline	88.8	16.5	79.3	94.5
TiedResNeXt101-S	**64.0** **(65%)**	**14.6** **(78%)**	79.3	94.5
SENet101 (Hu, Shen, and Sun 2018)				
baseline	49.1	7.9	77.6	93.9
baseline ‡	49.1	7.9	78.3	94.2
TiedSENet101-S	**26.4** **(54%)**	**5.2** **(66%)**	79.0	94.5
TiedSENet101-S †	**26.4** **(54%)**	**5.2** **(66%)**	**80.9**	**95.3**
TiedSENet101	41.8 (85%)	9.1 (115%)	79.8	94.8
ResNeSt-50-fast (Zhang et al. 2020)				
baseline ‡	27.5	4.4	78.6	93.9
TiedResNeSt50-S	**16.5** **(60%)**	**3.6** **(82%)**	**78.8**	**94.6**
VS. pruning methods and Mobile nets (large model version)				
Taylor-FO-BN	14.2	2.3	74.5	-
ShuffleNet-50 †	-	2.3	74.8	-
GhostNet-50 (s=2)	**13.0**	**2.2**	75.0	92.3
TiedResNet50-S	13.9	2.5	**76.2**	**92.9**

Table 1: **Recognition accuracy and model size comparison on ImageNet-1k**. The integration of TBC/TFC/TGC can obtain consistent performance improvements to various backbone networks. TiedResNet-S even greatly surpasses current SOTA pruning methods Taylor-FO-BN-ResNet50 (Molchanov et al. 2019) and Mobile architecture GhostNet (large model version) (Han et al. 2020). These results prove that TBC makes more efficient use of parameters. Baselines are copied from Pytorch model zoo, their TBC versions are trained for 100 epochs on 8 2080Ti GPUs to make fair comparisons, unless otherwise noticed. † denotes: trained with larger epochs, label smoothing, cosine learning scheduler and heavier data augmentation. ‡ denotes: re-implemented results with released codes and 100 training epochs.

Object Detection and Instance Segmentation

MS-COCO (Lin et al. 2014) consists of 80 object categories with 118K/5K/208K images for training (*train-2017*), validation (*val-2017*) and testing (*test-2017*) respectively. Average Precision (AP) across IoU thresholds from 0.5 to 0.95 with an interval of 0.05 is evaluated. Detection performance at various qualities, AP_{50} and AP_{75}, and at different scales, AP_S, AP_M and AP_L, are reported. All models are trained on *train-2017* split and results reported on *val-2017*.

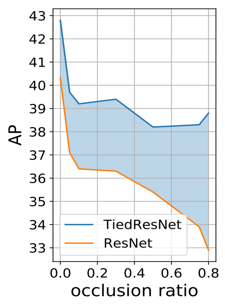
Implementation. We use baseline backbones and our TiedResNet model in PyTorch implemented (Chen et al. 2019) detectors. The long and short edges of images are resized to a maximum of 1333 and 800 respectively without changing the aspect ratio. Since 1× learning schedule (LS) is under-sutured, we only report results on 2× LS for both baselines and our models.

Results. We conduct thorough comparisons with ResNeXt and ResNet on multiple state-of-the-art frameworks including single-stage detector, RetinaNet (Lin et al. 2017), and two-stage detectors and Mask R-CNN (He et al. 2017) as in Fig.6. Since (Chen et al. 2019) re-implemented results are generally better than those in the original papers, we report re-implemented results for fair comparisons.

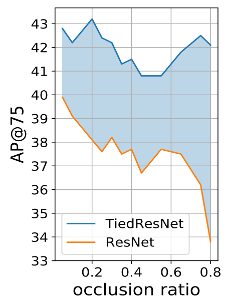
Object detection. As in Fig.6, using TiedResNet as backbone, single-stage detector RestinaNet and two-stage detector Cascade R-CNN and Mask R-CNN consistently outperform baselines by 2% to 2.5% in terms of box AP. TiedResNet101 on RetinaNet even greatly outperforms the much heavier-weight ResNeXt101-64×4d. Detailed comparison on various frameworks and Pascal VOC (Everingham et al. 2015) are in appendix materials.

Instance segmentation. With light-weight TiedResNet-S and comparable sized TiedResNet backbones, we observe an increase in AP^{mask} by 1.1% and 2.1% respectively. No matter how strong the baseline detector is, we always observe a boost in AP, corroborating the effectiveness of TBC.

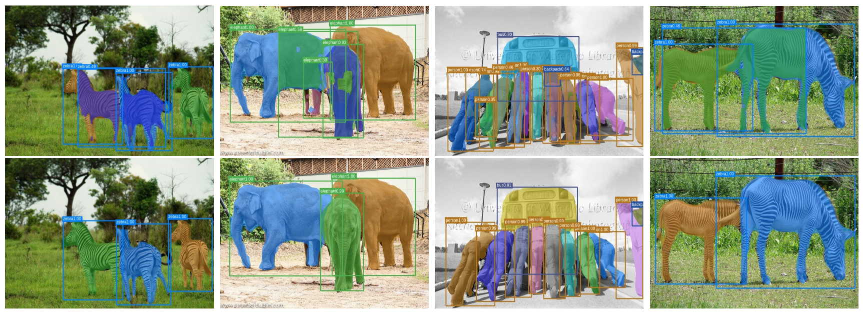
Highly occluded Instances. Since occlusion requires the network to accurately detect the target area and distinguish different instances at the same time, the performance on images with large occlusion reveals the network's localization



(a) r vs. AP



(b) r vs. AP^{75}



(c) Sample results at various occlusion ratios using ResNet (row 1) and TiedResNet (row 2)

Figure 7: **Our TiedResNet consistently outperforms ResNet on MS-COCO object detection under occlusion.** AP (a) and AP at IoU = 0.75 (b) are plotted against occlusion ratio r. When r = 0.8, TiedResNet increases by *8.3%* at AP^{75} and *5.9%* at AP, much more effective at handling highly overlapping instances. (c) TiedResNet has much fewer false positive proposals, and has a significantly better instance segmentation quality. We use Mask R-CNN as the detector.

framework	backbone	#params (M)	AP$^{\text{mask}}$
Mask R-CNN	ResNet50	25.6	31.5
Mask R-CNN	TiedResNet50-S	13.9	32.5
Mask R-CNN	TiedResNet50	22.0	**33.6**

Table 2: Comparison on **instance segmentation task of Cityscapes** *val* set and number of parameters for backbone networks, with Mask R-CNN (He et al. 2017) as detector.

capabilities. The occlusion ratio (r) of each image is:

$$r = \frac{\text{total overlap area}}{\text{total instance area}} \tag{5}$$

The AP averaged over IoU 0.5 to 0.95, and at IoU=0.75, AP^{75}, are used as standard and restricted evaluation metrics respectively. Fig.7a and Fig.7b shows that ResNet is greatly affected by occlusion, AP^{75} drops by more than 6% at $r = 0.8$, whereas our TiedResNet only slightly decreases by 0.7%, exceeding the baseline of *8.3%*. Similarly, as the occlusion rate becomes larger, the improvement on AP increases from 2.8% to *5.9%*. These quantitative results in MS-COCO indicate TiedResNet's strong capability of handling highly overlapping instances, especially on restricted evaluation metric. Fig.7c shows that TiedResNet has fewer false positive proposals and better segmentation quality.

Why larger gain on single-stage detector? Fig.8 shows that TiedResNet localizes the target area much better than ResNet/ResNeXt, which is especially beneficial for a single-stage detector that does not has a proposal regression layer.

Performance on Cityscapes. Since Cityscapes (Cordts et al. 2016) is a small dataset, thus deeper networks will generally overfit it. Therefore, we only deploy experiments with 50 layers backbone for Cityscapes datasets. Table 2 shows that TiedResNet50 can reach 2.1% gain for AP$^{\text{mask}}$.

Lightweight Attention

Fig. 5 shows our lightweight attention modules. The SE module can be seen as a special case of our TiedSE when $B=1$; likewise, GCB is TiedGCB at $B=1$.

Results of TiedSE. All experiments in Table 3 use reduction ratio of 16 for both baseline and our model. Several hyper-parameter settings of our TFC layer are investigated. Since

model	B	top-1 (%)	top-5 (%)	#params (ratio)
SEResNet-50, model params = 28.1M				
w/ SE	-	76.71	93.38	2.53M (100%)
w/ SE ‡	-	77.08	93.51	2.53M (100%)
w/ TiedSE	2	77.07	**93.53**	0.64M (25%)
w/ TiedSE	4	**77.11**	93.52	0.16M (6.4%)
w/ TiedSE	8	77.09	93.52	**0.04M (1.6%)**
EfficientNet-B0, model params = 5.3M				
w/ SE	-	77.1	93.3	0.65M (100%)
w/ TiedSE	2	**77.3**	**93.4**	0.16M (25%)
w/ TiedSE	4	77.1	93.3	**0.04M (6.4%)**

Table 3: Using only 1.6% (6.4%) of the parameters, the performance of TiedSE is better than SE on SERes-Net50 (EfficientNet-B0). We compared #params of **attention module SE/TiedSE** with various backbones and their recognition accuracy on ImageNet-1k. Performance with different hyper-parameters B is investigated. ‡ denotes our re-implementation results.

our re-implemented baseline results are better than those in (Hu, Shen, and Sun 2018), we report our results for fair comparison. While SE is light weight, it still incurs 10% parameters of overall model. Table 3 shows that, at B=8, with 64× parameters reduction, TiedSE still obtains comparable performance. TiedSE significantly reduces parameters without sacrificing performance not only on SEResNet but also on Mobile architecture EfficientNet (Tan and Le 2019).

Results of TiedGCB. Global context blocks (GCB) (Cao et al. 2019) enhance segmentation and detection predictions with global context modeling and long-range dependencies. GCB integrated with TBC can significantly reduce the number of parameters without losing performance. Table 4 shows that TiedGCB achieves 1.8% and 1.4% gain in AP$^{\text{mask}}$ and AP$^{\text{bbox}}$ respectively, with 16× parameters reduction. Although group convolution can reduce parameters by 2×, as each GC filter only sees a subset of features, the ability to model cross-channel dependencies is also reduced, losing AP$^{\text{mask}}$ and AP$^{\text{bbox}}$ by 0.4%.

framework	B	AP^{bbox}	AP^{bbox}_{50}	AP^{mask}	AP^{mask}_{50}	#params
Mask R-CNN	-	37.3	59.0	34.2	55.9	-
+GCB	-	38.9	**61.0**	35.5	**57.6**	10M (100%)
+TiedGCB	2	**39.1**	**61.0**	**35.6**	**57.6**	2.5M (25%)
+TiedGCB	4	38.6	60.8	35.2	57.2	**1.3M (13%)**

Table 4: Comparison on #params of **attention module GCB/TiedGCB** (Cao et al. 2019) and their performance on object detection and instance segmentation tasks of MS-COCO *val-2017*. The effects of different B are studied here.

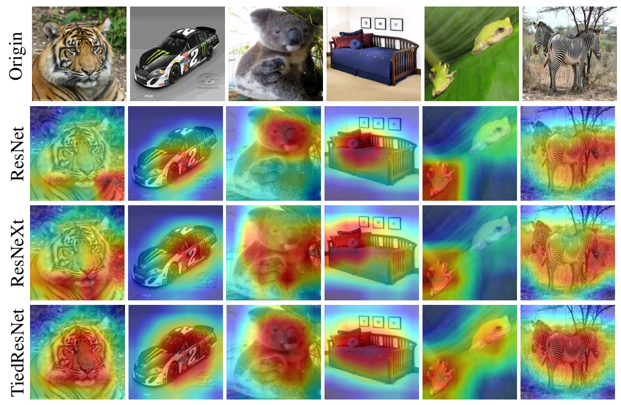


Figure 8: TiedResNet focusing on target objects more properly than ResNet and ResNeXt. We compared **Grad-CAM visualization** among ResNet50, ResNeXt50 and TiedResNet50 for images in Row 1. The grad-CAM (Selvaraju et al. 2017) is calculated for the last convolutional output.

model	setting	params	GFlops	top-1	top-5
TiedResNet-50	2s×48w	23.8	4.4	77.27	93.53
TiedResNet-50	4s×32w	**22.0**	4.4	**77.61**	**93.62**
TiedResNet-50	6s×24w	23.0	4.6	77.37	93.66
TiedResNet-50	8s×18w	23.8	4.4	77.21	93.54

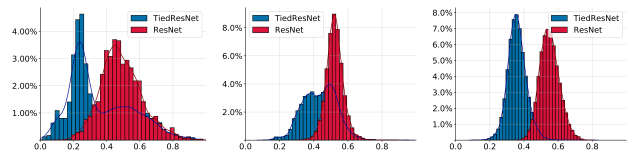
Table 5: Ablation study on **splits number** and **base width of each split**. Accuracies (%) on ImageNet-1k are listed.

mixer	top-1 acc.	top-5 acc.	#params (M)
element-sum	77.61%	93.62%	**22.0**
concatenate	**77.65%**	**93.64%**	26.7

Table 6: Ablation study on **fusion method** of mixer module.

Ablation Studies

Influence of split number. As investigated in (Zeiler and Fergus 2014; Bau et al. 2017; Xu et al. 2015), the proportions of units/filters that correspond to various visual concepts, such as color, texture, objects, part, scene, edge and material, are different with a variety of levels of interpretability (Agrawal, Carreira, and Malik 2015; Bau et al. 2017). It may be useful to group different functional filters together for different levels of sharing. In Table 5, we split all the channels in the 3×3 convolutional layer into s splits. Each split has base width of w, and B is 1,2,4,8 separately for the four 3×3 TBC layers in $4s \times 32w$ settings. In Table 5, the best performance and model complexity trade-off can be reached at $4s \times 32w$. Table 5 also shows the necessity of splitting input feature maps into several chunks, when there are only 2 splits, top-1 accuracy will drop 0.4%.



(a) Layer 9 (b) Layer 18 (c) Layer 36

Figure 9: Our TiedResNet learns less correlated filters than ResNet, with filter similarity histograms shifted left.

Mixer module in TiedBottleneck. Since we split the input feature map into several splits, the inter-dependency across these splits is missed. To track the inter-dependency, a mixer is used to aggregate cross-split information. Several fusion methods are investigated in Table 6. Using concatenation reaches the best accuracy, but it introduces much more parameters. We thus choose elementwise-sum as the fusion function as a trade-off between accuracy and model size.

Filter similarity. We use ImageNet pre-trained ResNet50 and TiedResNet50-S to compare the cosine filter similarity at different layers. Pairwise cosine similarity between filters' guided back-propagation patterns (Springenberg et al. 2014) averaged in 1000 ImageNet *val* split are used to generate these histograms. As in Figure 9, axis x is the cosine similarity and axis y is the probability density. Compared with VGG(Simonyan and Zisserman 2014), ResNet(He et al. 2016) has less redundancy, and our TiedResNet has the least similarity and thus removes most redundancy throughout the depth layers, which validates our hypothesis and motivation.

Grad-CAM visualization. To provide a qualitative comparison among different backbone networks, we apply grad-CAM (Selvaraju et al. 2017) using images from ImageNet. Grad-CAM uses the gradient information flowing into the last convolutional layer of the CNN to understand each neuron. The resulting localization map highlights important regions in the image for predicting the concept and reflects the network's ability to utilize information in the target object area. Fig.8 shows TiedResNet focusing on target objects more properly than ResNet and ResNetX, suggesting that the performance boost comes from accurate attention and noise reduction of irrelevant clutters.

This property is very useful for object detection and instance segmentation, as these tasks require the network to focus more accurately on the target region and aggregate features from it. Incorrect attention to the target area would also lead to a large number of false positive proposals (Fig.7c).

Summary

We propose Tied Block Convolution (TBC) that produces multiple responses with a single thinner filter shared across equal blocks of channels. This concept is extended to group convolution and fully connected layer function, and applied on various backbone networks and attention modules, with consistent accuracy gain and model reduction. TBC reduces filter redundancy in an optimized CNN and effectively expands model expression, resulting in better object detection and segmentation especially with occlusion.

Acknowledgements. This research was supported, in part, by Berkeley Deep Drive and DARPA.

References

Agrawal, P.; Carreira, J.; and Malik, J. 2015. Learning to see by moving. In *Proceedings of the IEEE International Conference on Computer Vision*, 37–45.

Bau, D.; Zhou, B.; Khosla, A.; Oliva, A.; and Torralba, A. 2017. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6541–6549.

Cao, Y.; Xu, J.; Lin, S.; Wei, F.; and Hu, H. 2019. GC-Net: Non-local Networks Meet Squeeze-Excitation Networks and Beyond. *arXiv preprint arXiv:1904.11492* .

Carreira, J.; and Zisserman, A. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6299–6308.

Chen, K.; Wang, J.; Pang, J.; Cao, Y.; Xiong, Y.; Li, X.; Sun, S.; Feng, W.; Liu, Z.; Xu, J.; Zhang, Z.; Cheng, D.; Zhu, C.; Cheng, T.; Zhao, Q.; Li, B.; Lu, X.; Zhu, R.; Wu, Y.; Dai, J.; Wang, J.; Shi, J.; Ouyang, W.; Loy, C. C.; and Lin, D. 2019. MMDetection: Open MMLab Detection Toolbox and Benchmark. *arXiv preprint arXiv:1906.07155* .

Chen, L.-C.; Papandreou, G.; Kokkinos, I.; Murphy, K.; and Yuille, A. L. 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence* 40(4): 834–848.

Chen, L.-C.; Zhu, Y.; Papandreou, G.; Schroff, F.; and Adam, H. 2018. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)*, 801–818.

Cordts, M.; Omran, M.; Ramos, S.; Rehfeld, T.; Enzweiler, M.; Benenson, R.; Franke, U.; Roth, S.; and Schiele, B. 2016. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3213–3223.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, 248–255. Ieee.

Everingham, M.; Eslami, S. M. A.; Van Gool, L.; Williams, C. K. I.; Winn, J.; and Zisserman, A. 2015. The Pascal Visual Object Classes Challenge: A Retrospective. *International Journal of Computer Vision* 111(1): 98–136.

Girshick, R. 2015. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 1440–1448.

Han, K.; Wang, Y.; Tian, Q.; Guo, J.; Xu, C.; and Xu, C. 2020. GhostNet: More features from cheap operations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1580–1589.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Howard, A. G.; Zhu, M.; Chen, B.; Kalenichenko, D.; Wang, W.; Weyand, T.; Andreetto, M.; and Adam, H. 2017. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861* .

Hu, J.; Shen, L.; and Sun, G. 2018. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7132–7141.

Ji, S.; Xu, W.; Yang, M.; and Yu, K. 2012. 3D convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence* 35(1): 221–231.

Kirillov, A.; Girshick, R.; He, K.; and Dollár, P. 2019. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6399–6408.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 1097–1105.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, 2980–2988.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*, 740–755. Springer.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Ma, N.; Zhang, X.; Zheng, H.-T.; and Sun, J. 2018. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 116–131.

Molchanov, P.; Mallya, A.; Tyree, S.; Frosio, I.; and Kautz, J. 2019. Importance estimation for neural network pruning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 11264–11272.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 91–99.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, 618–626.

Simonyan, K.; and Zisserman, A. 2014. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556* .

Springenberg, J. T.; Dosovitskiy, A.; Brox, T.; and Riedmiller, M. 2014. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806* .

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S.; Anguelov, D.; Erhan, D.; Vanhoucke, V.; and Rabinovich, A. 2015. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1–9.

Tan, M.; and Le, Q. V. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946* .

Wang, J.; Sun, K.; Cheng, T.; Jiang, B.; Deng, C.; Zhao, Y.; Liu, D.; Mu, Y.; Tan, M.; Wang, X.; et al. 2019. Deep high-resolution representation learning for visual recognition. *arXiv preprint arXiv:1908.07919* .

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Van Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *European conference on computer vision*, 20–36. Springer.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7794–7803.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, 2048–2057.

Zeiler, M. D.; and Fergus, R. 2014. Visualizing and understanding convolutional networks. In *European conference on computer vision*, 818–833. Springer.

Zhang, H.; Wu, C.; Zhang, Z.; Zhu, Y.; Zhang, Z.; Lin, H.; Sun, Y.; He, T.; Mueller, J.; Manmatha, R.; et al. 2020. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955* .

Zhang, X.; Zhou, X.; Lin, M.; and Sun, J. 2018. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6848–6856.