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

Xudong Wang, 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 $64 \times$ 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.

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
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.

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 variety 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 MSCOCO, 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) outperforms 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 adaptively 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 \( X \in \mathbb{R}^{c \times h \times w} \) and \( \tilde{X} \in \mathbb{R}^{c' \times h' \times w'} \) 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 \( \ast \), can be formulated as:

\[
\tilde{X} = X \ast W
\]  
(1)

where \( W \in \mathbb{R}^{c' \times k \times k} \) is the SC kernel. The number of parameters for SC is thus: \( c_0 \times c_1 \times k \times k \).

**Group Convolution** first divides input feature \( X \) into \( G \) equal-sized groups \( X_1, ..., X_G \) with size \( G \times h \times w \) per group. Each group shares the same convolutional filters \( W_g \). The output of GC is computed as:

\[
\tilde{X} = X_1 \ast W_1 \oplus X_2 \ast W_2 \oplus \cdots \oplus X_G \ast W_G
\]  
(2)

where \( \oplus \) is the concatenation operation along the channel dimension, \( W_g \) is the convolution filters for group \( g \), where \( g \in \{1, \ldots, G\} \), \( W_g \in \mathbb{R}^{c_0 \times k \times k} \). The number of parameters for GC is: \( G \times c_0 \times k \times k \).

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

\[
\tilde{X} = X_1 \ast W' \oplus X_2 \ast W' \oplus \cdots \oplus X_B \ast W'
\]  
(3)

where \( W' \in \mathbb{R}^{c_0 \times k} \) is the TBC filters shared among all the groups. The parameter number is: \( \frac{c_0}{B} \times B \times k \times k \).

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

**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× model size reduction, demonstrating that TiedResNet makes more effective use of model parameters.

**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.

10229
The default setting for TiedResNet-50 (TiedResNeXt-50) is 4 splits with base width of 32 (64), i.e., 4x×32w (4x×64w), and the default setting for TiedResNet-S (TiedResNeXt-50-S) is 4x×18w (4x×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**

**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 55% of parameters and 82% of computation cost, TiedResNet50-S obtains better performance than ResNeSt50-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.

### 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<sub>50</sub> and AP<sub>75</sub>, and at different scales, AP<sub>S</sub>, AP<sub>M</sub> and AP<sub>L</sub>, 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. 2020) 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-satured, 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. TiedResNet-101 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<sub>mask</sub> 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.

---

**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.

---

**Figure 6:** Chart showing the performance comparison of different models on MS-COCO val-2017. The x-axis represents the number of parameters (M), and the y-axis represents the Average Precision (AP) of models. The models are categorized into four groups: ResNet50, ResNet101, ResNeXt, and HRNetV2. Each group includes baseline models and their TBC versions. The chart illustrates how TBC makes more efficient use of parameters and improves performance consistently across various models.
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.

<table>
<thead>
<tr>
<th>framework</th>
<th>backbone</th>
<th>#params (M)</th>
<th>AP$^5_{mask}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mask R-CNN</td>
<td>ResNet50</td>
<td>25.6</td>
<td>31.5</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>TiedResNet50-S</td>
<td>13.9</td>
<td>32.5</td>
</tr>
<tr>
<td>Mask R-CNN</td>
<td>TiedResNet50</td>
<td>22.0</td>
<td>33.6</td>
</tr>
</tbody>
</table>

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.

The occlusion ratio ($r$) of each image is:

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

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 have 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$^5_{mask}$.

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 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 $64 \times$ 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$^5_{mask}$ and AP$^5_{bbox}$ respectively, with $16 \times$ parameters reduction. Although group convolution can reduce parameters by $2 \times$, as each GC filter only sees a subset of features, the ability to model cross-channel dependencies is also reduced, losing AP$^5_{mask}$ and AP$^5_{bbox}$ by 0.4%.

Table 3: Using only 1.6% (6.4%) of the parameters, the performance of TiedSE is better than SE on SEResNet50 (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.

<table>
<thead>
<tr>
<th>model</th>
<th>$B$</th>
<th>top-1 (%)</th>
<th>top-5 (%)</th>
<th>#params (ratio)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEResNet-50, model params = 28.1M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ SE</td>
<td>-</td>
<td>76.71</td>
<td>93.38</td>
<td>2.53M (100%)</td>
</tr>
<tr>
<td>w/ SE ‡</td>
<td>-</td>
<td>77.08</td>
<td>93.51</td>
<td>2.53M (100%)</td>
</tr>
<tr>
<td>w/ TiedSE</td>
<td>2</td>
<td>77.07</td>
<td>93.53</td>
<td>0.64M (25%)</td>
</tr>
<tr>
<td>w/ TiedSE</td>
<td>4</td>
<td><strong>77.11</strong></td>
<td>93.52</td>
<td>0.16M (6.4%)</td>
</tr>
<tr>
<td>w/ TiedSE</td>
<td>8</td>
<td>77.09</td>
<td>93.52</td>
<td><strong>0.04M (1.6%)</strong></td>
</tr>
<tr>
<td>EfficientNet-B0, model params = 5.3M</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>w/ SE</td>
<td>-</td>
<td>77.1</td>
<td>93.3</td>
<td>0.65M (100%)</td>
</tr>
<tr>
<td>w/ TiedSE</td>
<td>2</td>
<td><strong>77.3</strong></td>
<td><strong>93.4</strong></td>
<td>0.16M (25%)</td>
</tr>
<tr>
<td>w/ TiedSE</td>
<td>4</td>
<td>77.1</td>
<td>93.3</td>
<td><strong>0.04M (6.4%)</strong></td>
</tr>
</tbody>
</table>
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 $3 \times 3$ convolutional layer into $s$ splits. Each split has base width of $w$, and $B$ is 1,2,4,8 separately for the four $3 \times 3$ 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%.

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. 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 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.

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

<table>
<thead>
<tr>
<th>mixer</th>
<th>top-1 acc.</th>
<th>top-5 acc.</th>
<th>#params (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>element-sum</td>
<td>77.61%</td>
<td>93.62%</td>
<td>22.0</td>
</tr>
<tr>
<td>concatenate</td>
<td>77.65%</td>
<td>93.64%</td>
<td>26.7</td>
</tr>
</tbody>
</table>

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

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 by Berkeley Deep Drive and Defense Advanced Research Projects Agency (DARPA).

References


