GATED CHANNEL TRANSFORMATION FOR VISUAL RECOGNITION

Anonymous authors
Paper under double-blind review

ABSTRACT

In this work, we propose a generally applicable transformation unit for visual recognition with deep convolutional neural networks. This transformation explicitly models channel relationships with explainable control variables. These variables determine the neuron behaviors of competition or cooperation, and they are jointly optimized with convolutional weights towards more accurate recognition. In Squeeze-and-Excitation (SE) Networks, the channel relationships are implicitly learned by fully connected layers, and the SE block is integrated at the block-level. We instead introduce a channel normalization layer to reduce the number of parameters and computational complexity. This lightweight layer incorporates a simple $l_2$ normalization, enabling our transformation unit applicable to operator-level without much increase of additional parameters. Extensive experiments demonstrate the effectiveness of our unit with clear margins on many vision tasks, i.e., image classification on ImageNet, object detection and instance segmentation on COCO, video classification on Kinetics.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have proven to be critical and robust in visual recognition tasks, such as image classification (Huang et al., 2018), detection (Singh et al., 2018), and segmentation (Singh et al., 2018). Notably, a single convolutional layer operates only on a neighboring local context of each spatial position of a feature map, which could possibly lead to local ambiguities (Torralba, 2003). To relieve this problem, VGGNets (Simonyan & Zisserman, 2015) were proposed to construct deep CNNs, using a series of convolutional layers with non-linear activation functions and downsampling operators to cover a large extent of context. Moreover, He et al. (2016a) introduced a residual connection to help CNNs benefit from deeper architectures further.

Apart from improving the depth of CNNs, another branch of methods focuses on augmenting convolutional layer with modules that directly operate on context across large neighborhoods. Squeeze-and-Excitation Networks (SE-Nets) (Hu et al., 2018b) leveraged globally embedding information to model channel relationship and modulate feature maps on the channel-wise level. Moreover, its following method, GE-Nets (Hu et al., 2018a), used largely neighboring embedding instead. These modules can be conveniently assembled into modern networks, such as ResNets (He et al., 2016a) and Inception (Szegedy et al., 2015) networks, to improve the representational ability of networks.

However, the SE module uses two fully connected (FC) layers to process channel-wise embeddings, which leads to two problems. First, the number of SE modules to be applied in CNNs is limited. In Hu et al. (2018b), SE module was applied at the block-level, i.e., a single SE module is utilized per Res-block (He et al., 2016a) or Inception-block (Szegedy et al., 2016). The dimension of the FC layer is decreased to save the computational cost further. However, the designed FC layers still hinder the wide deployment of SE modules across all layers. Second, due to the complexity of the parameters in FC (or convolutional layer in GE), it is difficult to analyze the interactions among the channels at different layers. The channel relationships learned by convolution and FC operations are inherently implicit (Hu et al., 2018b), resulting in agnostic behaviors of the neuron outputs.

In this paper, we propose a Gated Channel Transformation (GCT) for efficient and accurate contextual information modeling. First, we use a normalization component to replace the FC layers. Normalization methods, e.g., Local Response Normalization (LRN) (Krizhevsky et al., 2012), create competitions among different neurons in neural networks. Batch normalization (Ioffe & Szegedy, 2015).
and its variants can smooth gradient and have been widely used in accelerating CNNs training process. We leverage a simple $l_2$ normalization for modeling channel relationship, which is more stable and computationally efficient comparing to FC layers. Second, we introduce a few channel-wise parameters to control the behavior of the gated adaptation of feature channels. Compared to the large number of parameters in FC, our designed parameters are much more lightweight. Besides, the gating weight parameter is convenient for channel relationship analysis and is helpful to understand the effect of GCT modules across different layers. According to our visualization analysis, GCT prefers to encourage cooperation in shallower layers, but competition is enhanced in deeper layers.

Our experiments show that GCT is a simple and effective architecture for modeling relationship among channels. It significantly improves the generalization capability of deep convolutional networks across visual recognition tasks and datasets.

2 RELATED WORK

Gating and attention mechanisms. Gating mechanisms have been successfully deployed in some recurrent neural network architectures. Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) introduced an input gate, output gate and forget gate, which are used to regulate the flow of information into and out of the module. Based on gating mechanisms, some attention methods focus on forcing computational resources towards the most informative components of features (Larochelle & Hinton, 2010; Mnih et al., 2014; Vaswani et al., 2017). Recent works introduce the attention mechanism into convolutional networks (e.g., Gehring et al., 2017; Dauphin et al., 2017). Following these studies, SE-Nets (Hu et al., 2018b) and its following work GE-Nets (Hu et al., 2018a) introduced a lightweight gating mechanism which focuses on enhancing the representational power of the convolutional network by modeling channel-wise relationship. Compared to the SE module, our GCT also pays attention to the cross-channel relationship but can achieve better performance gains with less computation and parameters.

Normalization layers. In recent years, normalization layers have been widely used in deep networks to create competition between neurons (Krizhevsky et al., 2012) and produce smoother optimization surfaces (Ioffe & Szegedy, 2015). Local Response Normalization (LRN) (Krizhevsky et al., 2012) computes the statistics in a small neighborhood among channels for each pixel. Batch Normalization (BN) (Ioffe & Szegedy, 2015) utilizes global spatial information along the batch dimension and suggests to be deployed for all layers. Layer Normalization (LN) (Ba et al., 2016) computes along the channel dimension instead of the batch dimension. Group Normalization (GN) (Wu & He, 2018) differently divides the channels into groups and computes within each group the mean and variance for normalization. Similar to LRN, GN and LN, our GCT also utilizes channel-related information with normalization structure.

Deep architectures. VGGNets (Simonyan & Zisserman, 2015) and Inception networks (Szegedy et al., 2015) demonstrated that it was significant to improve the quality of representation by increasing the depth of a network. ResNets (He et al., 2016a) utilized shortcut connections to identity-based skip connections, and proved that it was highly effective to build considerably deeper and stronger networks with them. Some other researchers focused on improving the representation ability of the computational elements contained within a network (Szegedy et al., 2016). The more diverse composition of operators within a computational element can be constructed with multi-branch convolutions or pooling layers. Other than this, grouped convolutions have proven to be a practical method to increase the cardinality of learned transformations (Xie et al., 2017). We build our GCT on these deep architectures. All these networks with GCT achieve promising performance improvements, but the growth of computational complexity is negligible.

3 GATED CHANNEL TRANSFORMATION

SE-Nets proposed a lightweight SE module to augment convolutional networks by operating on a global context. The SE module contains two operators, i.e., a “squeeze” operator to embed channel context and an “excitation” operator to modulate the feature maps. The architecture of our Gated Channel Transformation benefits from this framework. Differently, GCT leverages a normalization operator instead of the FC layers in the SE module for channel relationship modeling. Notably, the normalization operator is parameter-free. To make GCT learnable, we redesign the structure
Global Context Embedding

Channel Normalization

Gating Adaptation

Figure 1: An overview of the structure of Gated Channel Transformation (GCT).

of the “squeeze” and “excitation” operators. Our new operators contain three sets of channel-wise trainable parameters. Thus, GCT is more convenient to be deployed occupying a small number of parameters. The gating parameters can be visualized for easier analysis of GCT’s behavior, while the discriminative ability is maintained.

Let $x \in \mathbb{R}^{C \times H \times W}$ be an activation feature in a convolutional network, where $H$ and $W$ are the spatial height and width, and $C$ is the number of channels. In general, GCT performs the following transformation:

$$\hat{x} = F(x|\alpha, \gamma, \beta), \alpha, \gamma, \beta \in \mathbb{R}^C. \tag{1}$$

Here $\alpha$, $\gamma$ and $\beta$ are trainable parameters. Embedding weights $\alpha$ are responsible for adapting the embedding outputs. The gating weights $\gamma$ and biases $\beta$ control the activation of the gate. They determine the behavior of GCT in each channel. The parameter complexity of GCT is $O(C)$, which is smaller than the SE module ($O(C^2)$) (Hu et al., 2018b). In SE-Net, two FC layers are leveraged, which have the parameter complexity of $O(C^2)$.

An illustration of the structure of GCT is shown in Fig. 1. Let $x = [x_1, x_2, ..., x_C], x_c = [x_{c,i,j}]_{H \times W} \in \mathbb{R}^{H \times W}, c \in \{1, 2, ..., C\}$, where $x_c$ is corresponding to each channel of $x$. The detailed transformation consists of following parts.

**Global Context Embedding.** Global context embedding (GCE) aggregates global context in each channel. GCE can exploit global contextual information outside the small receptive fields of convolutional layers. Given the embedding weights $\alpha = [\alpha_1, ..., \alpha_C]$, GCE is defined as:

$$s_c = \alpha_c ||x_c||_2 = \alpha_c\left(\sum_{i=1}^{H} \sum_{j=1}^{W} (x_{c,i,j}^2) + \epsilon\right)^{1/2}, \tag{2}$$

where $\epsilon$ is a small constant to avoid the problem of derivation at the zero point. Different from SE, GCT does not use global average pooling (GAP) to aggregate channel context. GAP might fail in some extreme cases. For example, if SE is deployed after the Instance Normalization (Ulyanov et al., 2016) layer that is popular in style transfer task, the output of GAP will be constant for any inputs since IN fixes the mean of each channel of features. To avoid this problem, we choose $\ell_p$-norm instead. It is worth noting that GCT is robust with different $\ell_p$-norms. In Sec. 4.3, we compare the performance of some popular $\ell_p$-norms and choose the best one, $\ell_2$-norm, to be our default setting. Notably, the performance of $\ell_1$-norm is very close to $\ell_2$-norm and $\ell_1$-norm can be equivalently replaced by GAP when the input of GCT is always non-negative (for example, after ReLU activation). In this case, $\ell_1$-norm is more computationally efficient.

Besides, we use trainable parameters $\alpha_c$ to adjust each channel because different channels should have different significance.

**Channel Normalization.** Normalization methods can model relationship in visual or photographic features (Lyu & Simoncelli, 2008) with lightweight computing resource (e.g., Ioffe & Szegedy (2015)). Similar to LRN, we use a $\ell_2$ normalization to operate across channels, namely channel normalization (CN). Let $s = [s_1, ..., s_C]$, the formula of CN is:
where \( \epsilon \) is a small constant. The scalar \( \sqrt{C} \) is used to normalize the scale of \( \hat{s}_c \), avoiding a too small scale of \( \hat{s}_c \) when \( C \) is large. Compared to the FC layers used by SE, our CN operator has less computational complexity \( (O(C)) \) compared to the FC layers \( (O(C^2)) \).

**Gating Adaptation.** We employ a gating mechanism, namely gating adaptation, to adapt the original feature. By introducing the gating mechanism, our GCT can facilitate both competition and cooperation during the training process. Let the gating weights \( \gamma = [\gamma_1, ..., \gamma_C] \) and the gating biases \( \beta = [\beta_1, ..., \beta_C] \), we design the following gating function:

\[
\hat{x}_c = x_c [1 + \tanh(\gamma_c \hat{s}_c + \beta_c)].
\]

The scale of each original channel \( x_c \) will be adapted by its corresponding gate, i.e., \( 1 + \tanh(\gamma_c \hat{s}_c + \beta_c) \). The trainable \( \gamma_c \) and \( \beta_c \) are employed to control the activation of gate. LRN benefits from only the competitions among the neurons (Krizhevsky et al., 2012). However, the gating mechanism in GCT is able to create both competition and cooperation among different channels. This capability is more consistent with the training process in biological neural networks (Demin & Nekhaev, 2018). When the gating weight of one channel (\( \gamma_c \)) is activated positively, GCT promotes this channel to compete with the others as in LRN. When the gating weight is activated negatively, GCT encourages this channel to cooperate with the others. We analyze these adaptive channel relationships in Sec.4.4.

Besides, this gate function allows original features to pass to the next layer when the gating weights and biases are zeros, which is

\[
\hat{x} = x [1 + \tanh(\gamma_c \hat{s}_c + \beta_c)].
\]

The ability of modeling identity mapping can effectively improve the robustness to the degradation problem in deep networks. ResNets also benefit from this idea. Therefore, we propose to initialize \( \gamma \) and \( \beta \) to 0 in the initialization of GCT layers. By doing this, the initial steps of the training process will be more stable, and the final performance of GCT will be better.

## 4 Experiments

We apply GCT for all the convolutional layers in deep networks rather than block-level deployment in SE-Net. In all GCT counterparts, we employ one GCT layer before each convolutional layer. In the Kinetics experiments, we apply GCT at the last two convolutional layers in each Res-Block. More training details are shown in Appendix A.

### 4.1 Experiments on ImageNet

We experiment on the ImageNet 2012 dataset (Russakovsky et al., 2015) with 1,000 classes. We train all the models on the 1.28M training images and evaluate on the 50,000 validation images. We also conduct classification experiments on CIFAR (Krizhevsky & Hinton, 2009) in Appendix B.

**Implementation details.** In the training process of all the models, the input image is 224 × 224 randomly cropped from a resized image using the same augmentation in Szegedy et al. (2015). We
Table 1: Improvement in error performance (%) on ImageNet. The numbers in brackets denote the improvement in performance over the baselines. ResNet-200* means we follow the strategy in (He et al., 2016b) to train this model on $224 \times 224$ but evaluate on $320 \times 320$.

<table>
<thead>
<tr>
<th>Network</th>
<th>top-1</th>
<th>top-5</th>
<th>top-1</th>
<th>top-5</th>
</tr>
</thead>
<tbody>
<tr>
<td>VGG-16 (Simonyan &amp; Zisserman, 2015)</td>
<td>26.2</td>
<td>8.3</td>
<td>25.1</td>
<td>7.5</td>
</tr>
<tr>
<td>Inception-v3 (Szegedy et al., 2016)</td>
<td>24.3</td>
<td>7.3</td>
<td>23.7</td>
<td>7.1</td>
</tr>
<tr>
<td>ResNet-50 (He et al., 2016a)</td>
<td>22.4</td>
<td>6.3</td>
<td>21.7</td>
<td>6.0</td>
</tr>
<tr>
<td>ResNet-101 (He et al., 2016a)</td>
<td>23.8</td>
<td>7.0</td>
<td>22.7</td>
<td>6.3</td>
</tr>
<tr>
<td>ResNet-152 (He et al., 2016a)</td>
<td>22.2</td>
<td>6.2</td>
<td>21.4</td>
<td>5.9</td>
</tr>
<tr>
<td>ResNet-200* (He et al., 2016b)</td>
<td>21.6</td>
<td>5.9</td>
<td>20.8</td>
<td>5.5</td>
</tr>
<tr>
<td>ResNet-200 (He et al., 2016b)</td>
<td>20.7</td>
<td>5.2</td>
<td>19.7</td>
<td>4.8</td>
</tr>
</tbody>
</table>

Table 2: Compared to SE in different networks on ImageNet. We evaluate the models of error performance (%), GFLOPs (G) and parameters (M). G/P means GFLOPs/parameters. In VGG-16 experiments, SE is employed for all the convolutional layers, which is the same as GCT. In other experiments, SE is only employed in block level (Res-Block or Inception-Block) as proposed (Hu et al., 2018b). This difference makes that SE uses comparable GFLOPs with GCT.

<table>
<thead>
<tr>
<th>Network</th>
<th>original</th>
<th>SE</th>
<th>GCT (ours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>top-1/5 G/P</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ResNet-50</td>
<td>23.8/7.0</td>
<td>3.89/25.61</td>
<td>22.7/6.3</td>
</tr>
<tr>
<td>ResNet-50</td>
<td>22.4/6.3</td>
<td>3.79/25.10</td>
<td>21.7/5.9</td>
</tr>
<tr>
<td>VGG-16</td>
<td>26.2/8.3</td>
<td>15.49/138.37</td>
<td>25.1/7.5</td>
</tr>
<tr>
<td>Inception-v3</td>
<td>24.3/7.3</td>
<td>2.84/23.87</td>
<td>23.7/7.1</td>
</tr>
</tbody>
</table>

Integration with deep modern architectures. We study the effects of integrating GCT layers with some state-of-the-art backbone architectures, e.g., ResNets and ResNeXts (Xie et al., 2017), in which we apply GCT before all the convolutional layers. We report all these results in Table 1. Compared to original architectures, we observe significant performance improvements by introducing GCT into networks. Particularly, the top-1 error of GCT-ResNet-101 is 21.4%, which is even better than the ResNet-152 baseline (21.6%) with a deeper network and much more parameters. In addition, GCT is able to bring stable improvement in ResNets with different depth (1.1% top-1 improvement in ResNet-50, 0.8% in ResNet-152 and 1.0% in ResNet-200). Besides, we observe a smooth improvement throughout the training schedule, which is shown in Fig 2.

We also explore the improvement with GCT in non-residual networks (e.g., VGG-16 (Simonyan & Zisserman, 2015) and Inception-v3 (Szegedy et al., 2016)). To stabilize the training process, we employ BN (Ioffe & Szegedy, 2015) layers after every convolutional layer. Similar to the effectiveness in residual architectures, GCT layers bring promising improvements in non-residual structures.

Compared to SE. We conduct experiments on ImageNet to compare SE with GCT in both residual and non-residual networks and the results are reported in Table 2. We follow the methods in (Hu et al., 2018b) to integrate SE into VGG-16 (Simonyan & Zisserman, 2015), Inception (Szegedy et al., 2016), ResNet-50 (He et al., 2016a) and ResNeXt-50 (Xie et al., 2017) and train these models in same training schedule. Compared to SE, GCT always achieves better improvement.

In order to compare computational complexity, we calculate the GFLOPs and the number of parameters. In VGG-16 experiments, SE is employed for all the convolutional layers, which is the same as GCT. Under this fair condition, GCT achieves better performance with less increase in both GFLOPs (0.019G vs. 0.025G) and parameters (0.01M vs. 0.23M). In other experiments, SE is only employed in block-level (Res-Block or Inception-Block) as proposed (Hu et al., 2018b), which means the num-
Table 3: **Improvement on COCO with Mask R-CNN framework.** BN* means BN is frozen. + means increasing the training iterations from 90K to 270K. When using GN, we follow the strategy in the original paper (Wu & He [2018]).

<table>
<thead>
<tr>
<th>Backbone</th>
<th>box head</th>
<th>box AP</th>
<th>mask AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-50 BN*</td>
<td>-</td>
<td>37.8</td>
<td>34.2</td>
</tr>
<tr>
<td>ResNet-50 BN* +GCT</td>
<td>-</td>
<td>39.8 (2.0)</td>
<td>36.0 (1.8)</td>
</tr>
<tr>
<td>ResNet-101 BN*</td>
<td>-</td>
<td>40.1</td>
<td>36.1</td>
</tr>
<tr>
<td>ResNet-101 BN* +GCT</td>
<td>-</td>
<td>42.0 (1.9)</td>
<td>37.7 (1.6)</td>
</tr>
<tr>
<td>+ResNet-50 BN*</td>
<td>-</td>
<td>38.6</td>
<td>34.5</td>
</tr>
<tr>
<td>+ResNet-50 GN</td>
<td>GN</td>
<td>40.8 (2.2)</td>
<td>36.1 (1.6)</td>
</tr>
<tr>
<td>+ResNet-50 BN* +GCT</td>
<td>GN</td>
<td>41.6 (3.0)</td>
<td>37.1 (2.6)</td>
</tr>
<tr>
<td>+ResNet-50 BN* +GCT</td>
<td>GN +GCT</td>
<td>41.8 (3.2)</td>
<td>37.3 (2.8)</td>
</tr>
<tr>
<td>+ResNet-101 BN*</td>
<td>-</td>
<td>40.9</td>
<td>36.4</td>
</tr>
<tr>
<td>+ResNet-101 GN</td>
<td>GN</td>
<td>42.3 (1.4)</td>
<td>37.2 (0.8)</td>
</tr>
<tr>
<td>+ResNet-101 BN* +GCT</td>
<td>GN</td>
<td>43.1 (2.2)</td>
<td>38.3 (1.9)</td>
</tr>
</tbody>
</table>

The number of SE is smaller than GCT. However, the increase in parameters of GCT is still much less than SE, and the value of GFLOPs is comparable. Compared to SE, the increase in parameters of GCT is negligible, but the performance is better.

### 4.2 Experiments on COCO

Next we evaluate the generalizability on the COCO dataset (Lin et al. [2014]). We train the models on the COCO train2017 set and evaluate on the COCO eval2017 set (a.k.a minival).

**Implementation details.** We experiment on the Mask R-CNN baselines (He et al. [2017]) and its GN counterparts (Wu & He [2018]). All the backbone models are pre-trained on ImageNet using the scale and aspect ratio augmentation in Szegedy et al. [2015] and fine-tune on COCO with a batch size of 16 (2 images/GPU). Besides, all these experiments use the Feature Pyramid Network (FPN) Lin et al. [2017]. We also use the same hyperparameters and two training schedules used in Wu & He [2018]. The short schedule includes 90K iterations, in which the learning rate is divided by 10 at 60K and 80K iterations. The long schedule increases the iterations to 270K, in which the learning rate is divided by 10 at 210K and 250K. The base learning rate is 0.02 in both schedules.

**Improvements on Mask R-CNN.** Table 3 shows the comparison of BN* (frozen BN), GN and BN*+GCT (using GCT before all the convolutional layers of the backbones). First, we use a short training schedule to compare baselines and GCT counterparts. GCT shows stable and significant improvement in both ResNet-50 and ResNet-101. In ResNet-50, GCT improves detection AP by 2.0 and segmentation AP by 1.8. Moreover, in ResNet-101, GCT also improves detection AP by 1.9 and segmentation AP by 1.6. Then, we use the long schedule to compare GN and BN*+GCT. GN is more effective than BN when batch size is small as in this case of detection and segmentation using Mask R-CNN. However, we deploy GCT together with BN into the backbone, and these BN*+GCT counterparts achieve much better performance than GN backbones. Compared to GN in ResNet-101, BN*+GCT improves detection AP by 0.8 and segmentation AP by 1.1. In particular, ResNet-101 with BN*+GCT trained in the short schedule achieves better segmentation AP (37.7) than the GN counterpart (37.2) trained with the long schedule. This GN counterpart also uses GN in the backbone, the box heads, and the FPN. We also explore to combine GCT with GN by introducing GCT into GN box head. The results show GN+GCT achieves a better performance. It demonstrates the benefits of integrating GCT with GN. We now have shown the effectiveness of GCT in working with both BN and GN.

### 4.3 Experiments on Kinetics

Our previous experiments demonstrate the effectiveness of GCT on image-related tasks. We now evaluate the generalizability in video understanding task of action recognition on the large scale Kinetics-400 (Kay et al. [2017]) dataset. We employ the ResNet-50 (3D) and ResNet-101 (3D) as the backbone and apply GCT in the last two convolutional layers in each Res-Block. The backbone networks are pre-trained on ImageNet (Russakovsky et al. [2015]).
Figure 3: Analysis. The visualization of parameters of $\gamma$ (Fig.(a), (b)), and the ratio of variance of GCT output and input feature (Fig.(c)) in all the GCT layers in ResNet-50 on ImageNet.

We compare with the state-of-the-art Non-Local Networks (NL-Net) (Wang et al., 2018). The results show that GCT counterparts consistently improves the recognition accuracy over both the ResNet-50 and ResNet-101 baselines, as shown in Table 4. Because of our limited memory resource, we can NOT apply GCT in all the convolutional layers, which we believe can further improve the performance.

In summary, extensive experiments demonstrate that GCT is effective across a wide range of architectures, tasks, and datasets.

4.4 Analysis

To analyze the behavior of GCT in different layers, we visualize the distribution of the gating weights ($\gamma$) of each GCT layer in ResNet-50 on ImageNet. Further, we sort these distributions according to their layer index in 3D space (Fig. 3a). The bigger layer index means it is closer to the network output. To make the visualization clearer, we re-scale the vertical $z$ axis with $\log(1 + z)$, which corresponds to the percentage density of $\gamma$. We also calculate the mean and standard deviation (std) of $\gamma$ in each layer and show them in a bar chart (Fig. 3b). As shown in Fig. 3a and 3b, the mean of $\gamma$ tends to be less than 0 in the GCT layers far from the network output. Oppositely, in the layers close to the output, the mean tends to be greater than 0.

According to Eq. 3 & 4, the adaptation of channel $x_c$ is related to $\hat{s}_c$, which corresponds to the ratio of the weighted $\ell_2$-norm of $x_c$ (i.e., $s_c$) and the average of all the $s_c$. When the gating weight $\gamma_c$ is greater than 0, the adaptation is positively correlated to $\hat{s}_c$ and increases the variance between $x_c$ and others; When $\gamma_c$ is lower than 0, the adaptation is negatively correlated and reduces the variance.

Based on the analysis and the results we observe, we suppose that GCT tends to reduce the difference among channels in layers far away from the output. This behavior is helpful to encourage cooperation among channels and relieve overfitting. Apart from this, GCT tends to increase the difference among channels when close to the output. Here, GCT acts like the attention mechanism that is useful for creating competition.

To further validate our hypothesis, we calculate the ratio of the variance of output and input feature of each GCT layer, which we show in Fig. 3c. More visualizations are shown in Appendix C. Generally, the shallow layers learn low-level features to capture general characteristics like textures, edges, and corners. The feature variances become larger in deeper layers, where the high-level features are more discriminative and task-related. As expected, in the layers close to network output, GCT tends to magnify the variance of input feature (the ratio is always greater than 1), but in the layers far away from the output, GCT tends to reduce the variance (the ratio is always less than 1). This phenomenon is consistent with our previous hypothesis and shows that GCT is effective in creating both competition and cooperation among channels. Our observation validates that GCT can adaptively learn the channel relationships at different layers.

4.5 Ablation Studies.

In this section, we conduct a serial of ablation experiments to explain the relative importance of each operator in the GCT. At last, we show how the performance changes with regards to the GCT position in a network.
In this paper, we propose GCT, a novel layer that effectively improves the discriminability of deep CNNs by leveraging the relationship among channels. Benefit from the design of combining normalization and gating mechanisms, GCT can facilitate two types of neuron relations, i.e., competition and cooperation, with negligible complexity of parameters. We conduct expensive experiments to show the effectiveness and robustness of GCT across a wide range of modern CNNs and datasets. In future work, we will study the feasibility to apply GCT into recurrent networks.

5 Conclusion and Future Work

In this paper, we propose GCT, a novel layer that effectively improves the discriminability of deep CNNs by leveraging the relationship among channels. Benefit from the design of combining normalization and gating mechanisms, GCT can facilitate two types of neuron relations, i.e., competition and cooperation, with negligible complexity of parameters. We conduct expensive experiments to show the effectiveness and robustness of GCT across a wide range of modern CNNs and datasets. In future work, we will study the feasibility to apply GCT into recurrent networks.
REFERENCES


Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. Learning, 10:3.


## A Training details

Same to modern normalization layers (e.g., BN [Ioffe & Szegedy, 2015]), we propose to apply GCT for all convolutional layers in deep networks. However, there are many different points nearby one convolutional layer to employ GCT. In deep networks, each convolutional layer always works together with a normalization layer (e.g., BN [Ioffe & Szegedy, 2015]) and an activation layer (e.g., ReLU). For this reason, there are three possible points to deploy GCT layer, which are before the convolutional layer, before the normalization layer, and after the normalization layer. All these methods are effective, but we find to be better to employ GNC before the convolutional layer. In Sec.4.5 we compare the performance of these three application methods.

In the training process, we propose to use $1$ to initialize $\alpha$ and use $0$ to initialize all $\gamma$ and $\beta$. By doing this, GCT will be initialized as an identity mapping module, which will make the training process more stable. Besides, to avoid the bad influence of unstable gradient on the GCT gate in initial training steps, we propose to use warmup method (to start training with a small learning rate). In all the experiments on ImageNet (Russakovsky et al., 2015) and CIFAR (Krizhevsky & Hinton, 2009), we start training with a learning rate of $0.01$ for 1 epoch. After the warmup, we go back to the original learning rate schedule. Finally, we propose **NOT** to apply weight decay on $\beta$ parameters, which is possible to reduce the performance of GCT.


B EXPERIMENTS ON CIFAR

We conduct more experiments on the CIFAR-10 and CIFAR-100 datasets (Krizhevsky & Hinton, 2009). We follow the same training and testing strategies in (He et al., 2016a) to conduct our experiments but with a different learning rate schedule. We use a base learning rate of 0.1 and take cosine decay method (Loshchilov & Hutter) to adjust the learning rate for 300 epochs, which achieves better baseline performance. Following the above training protocol, we start the training process with a learning rate of 0.01 for 1 epoch.

We report the performance of ResNet-110 (He et al., 2016a) and its GCT counterpart in Table 7. To reduce the variances from different runs, we repeat all experiments for 5 times and report the averaged the results. As with the previous experiments, we observe promising improvements in performance, which shows that GCT can generalize to other image classification datasets.

Table 7: Improvement in top-1 error (%) on the CIFAR-10 and CIFAR-100 datasets.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>ResNet-110</th>
<th>GCT</th>
</tr>
</thead>
<tbody>
<tr>
<td>CIFAR-10</td>
<td>5.81</td>
<td>5.26 (0.55)</td>
</tr>
<tr>
<td>CIFAR-100</td>
<td>26.66</td>
<td>25.77 (0.89)</td>
</tr>
</tbody>
</table>

C MORE VISUALIZATION RESULTS

In ResNet-50 (He et al., 2016a) backbone, We visualize the channel activation before and after the GCT layer in both low-level stage (far away from the network output) and high-level stage (close to the output) in Fig.4 and 5, respectively. The input image is from the validation dataset of ImageNet. As we can see, for the stages far away from the network output, the proposed GCT layer tends to reduce the variance of input feature, which encourages cooperation among channels and avoids excessive activation values or loss of useful features. On the contrary, for those stages close to the output, GCT tends to magnify the variance. Here, GCT acts like the attention mechanism that is useful for creating competition.

Figure 4: Visualization of the channel activation for a GCT layer in Stage 2 (low-level) of ResNet-50 on the validation dataset of ImageNet.
Figure 5: Visualization of the channel activation for a GCT layer in Stage 5 (high-level) of ResNet-50 on the validation dataset of ImageNet.