CONTEXT-GATED CONVOLUTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As the basic building block of Convolutional Neural Networks (CNNs), the convolutional layer is designed to extract local patterns and lacks the ability to model global context in its nature. Many efforts have been recently made to complement CNNs with the global modeling ability, especially by a family of works on global feature interaction. In these works, the global context information is incorporated into local features before they are fed into convolutional layers. However, research on neuroscience reveals that, besides influences changing the inputs to our neurons, the neurons' ability of modifying their functions dynamically according to context is essential for perceptual tasks, which has been overlooked in most of CNNs. Motivated by this, we propose one novel Context-Gated Convolution (CGC) to explicitly modify the weights of convolutional layers adaptively under the guidance of global context. As such, being aware of the global context, the modulated convolution kernel of our proposed CGC can better extract representative local patterns and compose discriminative features. Moreover, our proposed CGC is lightweight, amenable to modern CNN architectures, and consistently improves the performance of CNNs according to extensive experiments on image classification, action recognition, and machine translation.

1 INTRODUCTION

Convolutional Neural Networks (CNNs) have achieved significant successes in various tasks, e.g., image classification (He et al., 2016a; Huang et al., 2017), object detection (Girshick et al., 2014; Ren et al., 2015), image translation (Zhu et al., 2017), action recognition (Carreira & Zisserman, 2017), sentence/text classification (Zhang et al., 2015; Kim, 2014), machine translation (Gehring et al., 2017), etc. However, the sliding window mechanism of convolution makes it only capable of capturing local patterns, limiting its ability of utilizing global context. Taking the 2D convolution on the image as one example, as Figure 1a shows, the standard convolution only operates on the local image patch and thereby composes local features.

According to the recent research on neuroscience (Gilbert & Li, 2013), neurons' awareness of global context is important for us to better interpret visual scenes, stably perceive objects and effectively process complex perceptual tasks. Many methods (Vaswani et al., 2017; Wang et al., 2017a;b; Park et al., 2018; Hu et al., 2018; Chen et al., 2019; Cao et al., 2019; Bello et al., 2019) have been
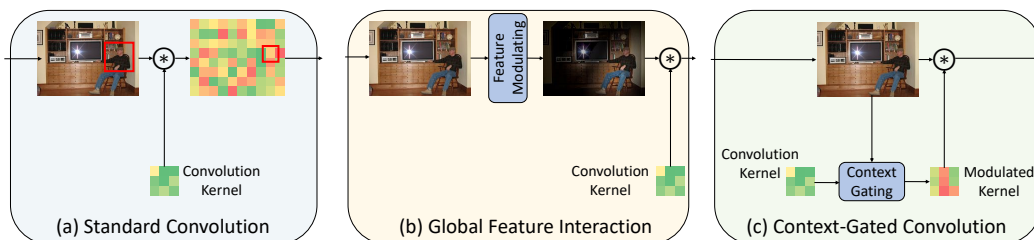

Figure 1: (a) Standard convolution only composes local information. (b) Global feature interaction methods modify input feature maps by incorporating global information. (c) Our CGC, in a fundamentally different manner, modulates convolution kernels under the guidance of global context. ⊛ denotes convolution.

proposed recently to introduce global context modeling modules into CNN architectures. In this paper, such a family of works is named as global feature interaction methods. As Figure 1b shows, these methods modulate intermediate feature maps by incorporating the global context with the local feature representation. For example, in Non-local modules (Wang et al., 2017b), local features are reassembled according to global correspondence, which augments CNNs with the global context modeling ability.

As was discussed by Gilbert & Li (2013), the global context influences neurons processing information in two distinct ways: "*various forms of attention, including spatial, object oriented and feature oriented attention*" and "*rather than having a fixed functional role, neurons are adaptive processors, changing their function according to behavioral context*". The previous work (Vaswani et al., 2017; Wang et al., 2017a;b; Park et al., 2018; Hu et al., 2018; Chen et al., 2019; Cao et al., 2019; Bello et al., 2019) of global feature interaction methods, shown in Figure 1b, only modifies intermediate features, namely, inputs of neurons, which corresponds to the first way. However, to the best of our knowledge, the other efficient and intuitive way, i.e., explicitly modulating the convolution kernels according to context, has not been exploited yet. Motivated by this, we will model convolutional layers as "*adaptive processors*" and explore how to leverage global context to guide the composition of local features in convolution operations.

In this paper, we propose Context-Gated Convolution (CGC), as Figure 1c shows, a new perspective of complementing CNNs with the awareness of the global context. Specifically, our proposed CGC learns a series of mappings to generate gates from the global context feature maps to modulate convolution kernels accordingly. With the modulated kernels, standard convolution is performed on input feature maps, which is enabled to dynamically capture representative local patterns and compose local features of interest under the guidance of global context. Our contributions are in three-fold.

- To the best of our knowledge, we make the first attempt of introducing the context-awareness to convolutional layers by modulating the weights of them according to the global context.

- We propose a novel lightweight CGC to effectively generate gates for convolution kernels to modify the weights with the guidance of global context. Our CGC consists of a Context Encoding Module that encodes context information into latent representations, a Channel Interacting Module that projects them into the space of output dimension, and a Gate Decoding Module that decodes the latent representations to produce the gate.

- Our Context-Gated Convolution can better capture local patterns and compose discriminative features, and consistently improve the performance of standard convolution with a negligible complexity increment in various tasks including image classification, action recognition, and machine translation.

2 CONTEXT-GATED CONVOLUTION

2.1 PRELIMINARY

Without loss of generality, we consider one sample of 2D case. The input to a convolutional layer is a feature map $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$, where c is the number of channels, and h, w are respectively the height and width of the feature map. In each convolution operation, a local patch of size $c \times k_1 \times k_2$ is collected by the sliding window to multiply with the kernel $\mathbf{W} \in \mathbb{R}^{o \times c \times k_1 \times k_2}$ of this convolutional layer, where o is the number of output channels, and k_1, k_2 are respectively the height and width of the kernel. Therefore, only local information within each patch is extracted in one convolution operation. Although in the training process, the convolution kernels are learned from all the patches from all the images in the training set, the kernels are not adaptive to the current context during inference time.

2.2 MODULE DESIGN

In order to handle the aforementioned drawback of standard convolution, we propose to incorporate the global context information during the convolution process. Different from the existing approaches that modify the input features according to the context, e.g., a global correspondence of
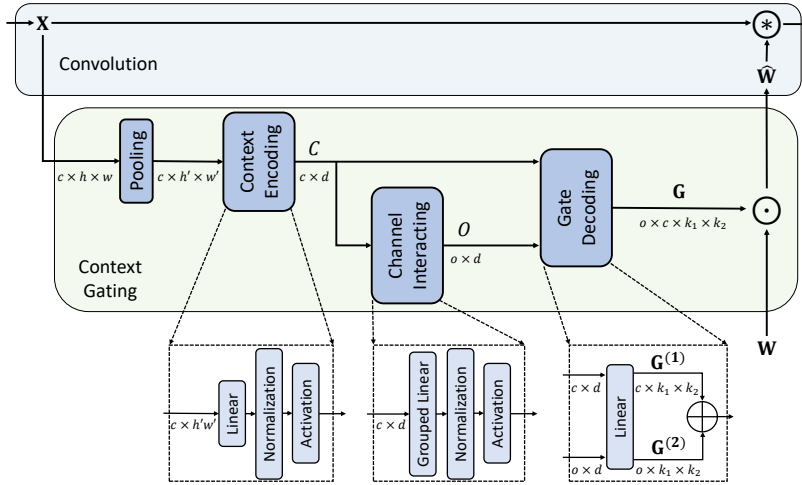
Figure 2: Our proposed CGC consists of three components, namely the Context Encoding Module, the Channel Interacting Module, and the Gate Decoding Module. The Context Encoding Module encodes global context information into a latent representation C; the Channel Interacting Module transforms C to O with output dimension o; the Gate Decoding Module produces $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ from C and O to construct the gate $\mathbf{G}$. $\circledast, \odot$ denote convolution and element-wise multiplication operations, respectively. $\oplus$ is shown in Equation 1.

feature representations, we attempt to directly modulate the convolution kernel under the guidance of the global context information.

One simple and straightforward way of modulating the convolution kernel $\mathbf{W}$ with global context information is to directly generate a gate $\mathbf{G} \in \mathbb{R}^{o \times c \times k_1 \times k_2}$ of the same size with $\mathbf{W}$ according to global context. Assuming that we generate the gate from a context vector $\boldsymbol{v} \in \mathbb{R}^l$ using a linear layer without the bias term, the number of parameters is $l \times o \times c \times k_1 \times k_2$, which is extremely catastrophic when we modulate the convolution kernel of every convolutional layer. For modern CNNs, o and c can be easily larger than 100 or even 1,000, which makes $o \times c$ the dominant term in the complexity. Inspired by previous works on convolution kernel decomposition (Howard et al., 2017; Chollet, 2017), we propose to decompose the gate $\mathbf{G}$ into two tensors $\mathbf{G}^{(1)} \in \mathbb{R}^{c \times k_1 \times k_2}$ and $\mathbf{G}^{(2)} \in \mathbb{R}^{o \times k_1 \times k_2}$, so that the complexity of $o \times c$ can thereby significantly break down.

However, directly generating these two tensors is still not acceptable. Supposing that we generate them with two linear layers, the number of parameters is $l \times (o + c) \times k_1 \times k_2$, which is at the same scale with the number of parameters of the convolution kernel itself. The bottleneck now is jointly modeling channel-wise and spatial interactions, namely l and $(o + c) \times k_1 \times k_2$, considering that $\boldsymbol{v} \in \mathbb{R}^l$ is encoded from the input feature map $\mathbf{X} \in \mathbb{R}^{c \times h \times w}$. Inspired by depth-wise separable convolutions (Howard et al., 2017; Chollet, 2017), we propose to model the spatial interaction and the channel-wise interaction separately to further reduce complexity.

In this paper, we propose one novel Context-Gated Convolution (CGC) to incorporate the global context information during the convolution process. Specifically, our proposed CGC consists of three modules: the Context Encoding Module, the Channel Interacting Module, and the Gate Decoding Module. As Figure 2 shows, the Context Encoding Module encodes global context information in each channel into a latent representation C via spatial interaction; the Channel Interacting Module projects the latent representation to the space of output dimension o via channel-wise interaction; the Gate Decoding Module produces $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ from the latent representation C and the projected representation O to construct the gate $\mathbf{G}$ via spatial interaction. The details of them are described in the following.

Context Encoding Module. To extract contextual information, we first use a pooling layer to reduce the spatial resolution to $h' \times w'$ and then feed the resized feature map to the Context Encoding

3

Module. It encodes information from all the spatial positions for each channel, and extracts a latent representation of the global context. We use a linear layer with weight $\mathbf{E} \in \mathbb{R}^{h' \times w' \times d}$ to project the resized feature map in each channel to a latent vector of size d. Inspired by the bottleneck structure from (He et al., 2016a; Hu et al., 2018; Wang et al., 2017b; Vaswani et al., 2017), we set $d = \frac{k_1 \times k_2}{2}$ to extract informative context, when not specified. The weight $\mathbf{E}$ is shared across different channels. A normalization layer and an activation function come after the linear layer. There are c channels, so the output of the Context Encoding Module is $\boldsymbol{C} \in \mathbb{R}^{c \times d}$.

Channel Interacting Module. It projects the feature representations $C \in \mathbb{R}^{c \times d}$ to the space of output dimension o. Inspired by (Ha et al., 2016), we use a grouped linear layer $\mathbf{I} \in \mathbb{R}^{\frac{c}{g} \times \frac{o}{g}}$, where g is the number of groups. The weight $\mathbf{I}$ is shared among different dimensions of d and different groups. A normalization layer and an activation function come after the linear layer. The final output of the Channel Interacting Module is $\boldsymbol{O} \in \mathbb{R}^{o \times d}$.

Gate Decoding Module. It takes both C and O as inputs, and decodes the latent representations to the spatial size of convolution kernels. We use two linear layers whose weights $\mathbf{D}_c \in \mathbb{R}^{d \times k_1 \times k_2}$ and $\mathbf{D}_o \in \mathbb{R}^{d \times k_1 \times k_2}$ are respectively shared across different channels in C and O. Then each element in the gate is produced by

$$\mathbf{G}_{h,i,j,k} = \sigma(\mathbf{G}_{i,j,k}^{(1)} + \mathbf{G}_{h,j,k}^{(2)}) = \sigma((C\mathbf{D})_{i,j,k} + (O\mathbf{D})_{h,j,k}), \tag{1}$$

where $\sigma(\cdot)$ denotes the sigmoid function. Now we have $\mathbf{G}$ with the same size of the convolution kernel $\mathbf{W}$, which is generated from the global context by our lightweight modules. Then we can modulate the weight of a convolutional layer by element-wise multiplication to incorporate rich context information:

$$\hat{\mathbf{W}} = \mathbf{W} \odot \mathbf{G}. \tag{2}$$

With the modulated kernel, a standard convolution process is performed on the input feature maps, where the context information can help the kernel capture more representative patterns and also compose features of interest.

Complexity. The computational complexity of our three modules is $O(c \times d \times h' \times w' + c \times o/g + c \times d \times k_1 \times k_2 + o \times d \times k_1 \times k_2 + o \times c \times k_1 \times k_2)$, where h', w' can be set independent of h, w. It is negligible compared to convolution's $O(o \times c \times k_1 \times k_2 \times h \times w)$. Except the linear time of pooling, the complexity of these three modules is independent of the input's spatial size. The total number of parameters is $O(d \times h' \times w' + c \times o/g^2 + 2 \times d \times k_1 \times k_2)$, which is negligible compared to standard convolution's $O(o \times c \times k_1 \times k_2)$. Therefore we can easily replace the standard convolution by our proposed CGC with a very limited computation and parameter increment, therefore enabling neurons to be adaptive to global context.

2.3 DISCUSSIONS

We are aware of previous works on dynamically modifying the convolution operation (Dai et al., 2017; Wu et al., 2019; Jia et al., 2016; Jo et al., 2018; Mildenhall et al., 2018). However, two key factors distinguish our approach from those works: whether the information guiding convolution is collected globally and how it changes the parameters of convolution. Dai et al. (2017) proposed to adaptively set the offset of each element in a convolution kernel, and Wu et al. (2019) proposed to dynamically generate the weights of convolution kernels. However, in their formulations, the dynamic mechanism for modifying convolution kernels only takes local patches or segments as inputs, so it is only adaptive to local inputs, which limits their ability of leveraging rich information in global context. According to experiments in Section 3.4, our proposed CGC significantly outperforms Dynamic Convolution (Wu et al., 2019) with the help of global context awareness.

Another family of works on dynamic filters (Jia et al., 2016; Jo et al., 2018; Mildenhall et al., 2018) generates weights of convolution kernels using features extracted from input images by another CNN feature extractor. The expensive feature extraction process makes it more suitable for generating a few filters, e.g., in the case of low-level image processing. It is impractical to generate weights for all the layers in a deep CNN model in this manner. However, our CGC takes input feature maps of a convolutional layer and makes it possible to dynamically modulate the weight of each convolutional layer, which systematically improves CNNs' global context modeling ability.

Table 1: Image classification results on ImageNet and CIFAR-10. Param indicates the number of parameters in the model. ΔMFLOPs is the increment of the number of multiplication-addition operations compared to ResNet-50 (4 GFLOPs) for ImageNet models and ResNet-110 (256 MFLOPs) for CIFAR-10 models. Bold indicates the best result.

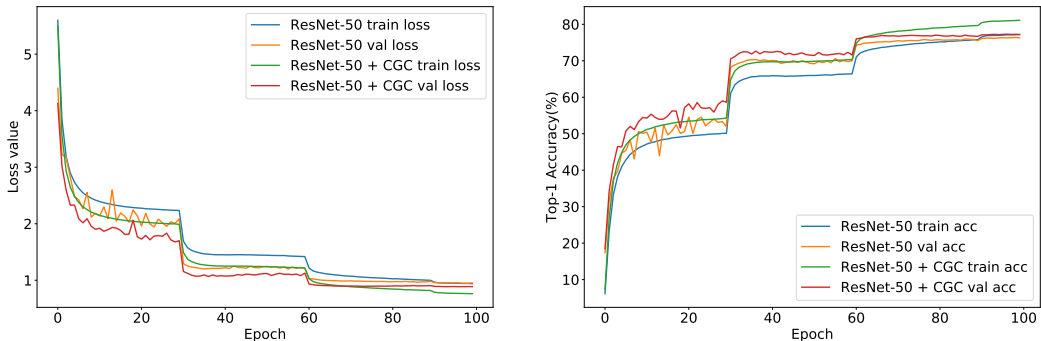| Dataset	Models	Param	ΔMFLOPs	Top-1(%)	Top-5(%)
ImageNet	SE-ResNet-50 (Hu et al., 2018)	28.09M	8	77.18	**93.70**
	BAM-ResNet-50 (Park et al., 2018)	25.92M	83	76.90	93.40
	GC-ResNet-50 (Cao et al., 2019)	28.11M	8	73.90	91.70
	ResNet-50 (He et al., 2016a)	25.56M	-	76.18	92.91
	ResNet-50 + CGC(Ours)	**25.59M**	**6**	**77.30**	93.66
CIFAR-10	ResNet-110 (He et al., 2016b)	1.73M	-	93.96	99.73
	ResNet-110 + CGC(Ours)	**1.80M**	**2**	**94.86**	**99.82**

Figure 3: The training curves of ResNet-50 and ResNet-50 + CGC (ours) on ImageNet.

3 EXPERIMENTS

In this section, we demonstrate the effectiveness of our proposed CGC in incorporating 1D, 2D, and 3D context information in 1D, 2D, and (2+1)D convolutions. We conduct extensive experiments on image classification, action recognition, and machine translation, and observe that our CGC consistently improves the performance of modern CNNs with negligible parameter increment on four benchmark datasets: ImageNet (Russakovsky et al., 2015), CIFAR-10 (Krizhevsky et al., 2009), Something-Something (v1) (Goyal et al., 2017), and IWSLT'14 De-En (Cettolo et al.).

3.1 IMPLEMENTATION DETAILS

All of the experiments are based on PyTorch (Paszke et al., 2017). All the linear layers are without bias terms. We follow common practice to use Batch Normalization (Ioffe & Szegedy, 2015) for computer vision tasks, or Layer Normalization (Ba et al., 2016) for natural language processing tasks, and ReLU (Nair & Hinton, 2010) as the activation function. Note that we learn different sets of scaling and shifting factors for C that is fed to the identity connection and for C that is fed to the Channel Interacting Module. We use average pooling with $h' = k_1$ and $w' = k_2$, when not specified. Note that we only replace convolution kernels with a spatial size larger than 1. For those Point-wise convolutions, we take them as linear layers and do not modulate them. To reduce the size of $\mathbf{l}$, we fix $c/g = 16$ when not specified. We initialize all these layers as what He et al. (2015) did for computer vision tasks and as what Glorot & Bengio (2010) did for natural language processing tasks.

3.2 IMAGE CLASSIFICATION

Experiment Setting. Following previous works (He et al., 2016a) on ImageNet, we train models on ImageNet 2012 training set, which contains about 1.28 million images from 1,000 categories,

Table 2: Ablation studies on CIFAR-10. Param indicates the number of parameters in the model. ΔMFLOPs is the increment of the number of multiplication-addition operations compared to ResNet-110 (256 MFLOPs). Bold indicates our default setting.

Models	Param	ΔMFLOPs	Top-1 Acc (%)
ResNet-110 (He et al., 2016b)	1.73M	-	93.96
ResNet-110 + CGC	1.80M	1.681	94.86
ResNet-110 + CGC only $\mathbf{G}^{(1)}$	1.75M	1.447	94.53
ResNet-110 + CGC only $\mathbf{G}^{(2)}$	1.78M	1.472	94.41
ResNet-110 + CGC $\mathbf{G}^{(1)} * \mathbf{G}^{(2)}$	1.80M	1.681	94.59
ResNet-110 + CGC $g = 1$	1.96M	1.681	94.97
ResNet-110 + CGC $d = k_1 + k_2$	1.81M	1.741	94.61
ResNet-110 + CGC Shared Norm	1.79M	1.681	94.72
ResNet-110 + CGC Two $\mathbf{E}$s	1.80M	1.871	94.53
ResNet-110 + CGC Shared $\mathbf{D}$	1.79M	1.681	94.78
ResNet-110 + CGC $h' = 2k_1, w' = 2k_2$	1.81M	1.681	94.68
ResNet-110 + CGC MaxPool	1.80M	1.681	94.44
ResNet-110 + CGC (res1,2,3)	1.80M	1.678	94.55
ResNet-110 + CGC (res2,3)	1.78M	1.052	94.43
ResNet-110 + CGC (res3)	1.76M	0.622	94.26

and report the results on its validation set, which contains 50,000 images. We replace all the 3×3 convolutions in ResNet-50 (He et al., 2016a) with our CGC and train the network from scratch. Note that for the first convolutional layer, we use $\mathbf{l} \in \mathbb{R}^{3 \times 64}$ for the Channel Interacting Module. We follow common practice (He et al., 2016a) and apply minimum training tricks to isolate the contribution of our CGC. CIFAR-10 contains 50K training images and 10K testing images in 10 classes. We follow common practice (He et al., 2016b) to train and evaluate the models. We take ResNet-110 (He et al., 2016b)(with plain blocks) as the baseline model and test other possibilities of generating the gate $\mathbf{G}$. All the compared methods are trained based on the same training protocol[1]. The details are provided in appendix. For evaluation, we report Top-1 and Top-5 accuracy of a single crop with the size 224×224 for ImageNet and 32×32 for CIFAR-10, respectively.

Performance Results. As Table 1 shows, our CGC significantly improves the performance of baseline models on both ImageNet and CIFAR-10. On ImageNet, our CGC improves the Top-1 accuracy of ResNet-50 by 1.12% with only 0.03M more parameters and 6M more FLOPs, which verifies our CGC's effectiveness of incorporating global context and its efficiency. We also find that GC-ResNet-50 is hard to train from scratch unless using the fine-tuning protocol reported by Cao et al. (2019), which indicates that modifying features may be misleading in the early training process. Although our CGC introduces a few new parameters, our model converges faster and more stably compared to vanilla ResNet-50, as is shown in Figure 3. We suppose that this is because the adaptiveness to global context improves the model's generalization ability and the gating mechanism reduces the norm of gradients back-propagated to the convolution kernels, which leads to a smaller Lipschitz constant and thus better training stability (Santurkar et al., 2018; Qiao et al., 2019).

Ablation Study. In order to demonstrate the effectiveness of our module design, ablation studies are conducted on CIFAR-10, as illustrated in Table 2. Specifically, we ablate many variants of our CGC and find our default setting a good trade-off between parameter increment and performance gain. The experiments on the combination of $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$ show that our decomposition approach in Equation 1 is a better way to construct the gate. For channel interacting, we find that using a full linear model with $g = 1$ achieves better performance with more parameters, as is expected. We try removing the bottleneck structure and set $d = k_1 + k_2$, and the performance drops, which validates the necessity of the bottleneck structure.

Shared Norm indicates learning the same set of scaling and shifting factors for C in the following two branches. Two $\mathbf{E}$s indicates that we learn another $\mathbf{E}$ to encode C only for the Channel Interacting Module. We also try sharing $\mathbf{D}$ for generating $\mathbf{G}^{(1)}$ and $\mathbf{G}^{(2)}$, using larger resized feature maps and

---

[1]The code is based on https://github.com/bearpaw/pytorch-classification

Table 3: Action recognition results on Something-Something (v1). Backbone indicates the backbone network architecture. Param indicates the number of parameters in the model. Frame indicates number of frames used for evaluation. Bold indicates the best result.

Models	Backbone	Param	Frame	Top-1(%)	Top-5(%)
TRN (Zhou et al., 2018)	BNInception	18.3M	8	34.4	-
TRN (Lin et al., 2018)	ResNet-50	31.8M	8	38.9	68.1
ECO (Zolfaghari et al., 2018)	BNInc+Res18	47.5M	8	39.6	-
ECO (Zolfaghari et al., 2018)	BNInc+Res18	47.5M	16	41.4	-
ECO_{En}Lite (Zolfaghari et al., 2018)	BNInc+Res18	150M	92	46.4	-
TSN (Wang et al., 2016)	ResNet-50	23.86M	8	19.00	44.98
TSN + Non-local (Wang et al., 2017b)	ResNet-50	31.22M	8	25.73	55.17
TSN + CGC (Ours)	ResNet-50	24.07M	8	32.58	60.06
P3D (Qiu et al., 2017)	ResNet-50	25.38M	32×30	45.17	74.61
P3D + Non-local (Wang et al., 2017b)	ResNet-50	32.73M	32×30	45.88	74.94
P3D + CGC 1D (Ours)	ResNet-50	25.39M	32×30	46.14	75.92
P3D + CGC 3D (Ours)	ResNet-50	25.61M	32×30	46.35	75.97
P3D + CGC 1D & 3D (Ours)	ResNet-50	25.62M	32×30	46.53	76.00
TSM (Lin et al., 2018)	ResNet-50	23.86M	8	44.65	73.94
TSM + Non-local (Wang et al., 2017b)	ResNet-50	31.22M	8	43.91	72.18
TSM + CGC (Ours)	ResNet-50	24.07M	8	46.00	75.11
TSM (Lin et al., 2018)	ResNet-50	23.86M	16	46.61	76.18
TSM + CGC (Ours)	ResNet-50	24.09M	16	**47.87**	**77.22**

using max pooling instead of average pooling. All the results support our default setting. We also test different numbers of layers to replace standard convolutions with our CGC. The result indicates that the more, the better.

3.3 ACTION RECOGNITION

Baseline Methods. For the action recognition task, we adapt three baselines to evaluate the effectiveness of our CGC: TSN (Wang et al., 2016), P3D-A (Qiu et al., 2017) (details are in appendix), and TSM (Lin et al., 2018). Because our CGC's effectiveness of introducing 2D spatial context to CNNs has been verified in image classification, in this part, we focus on its ability of incorporating 1D temporal context and 3D spatial-temporal context. For the 1D case, we apply our CGC to temporal convolutions in every P3D-A block. For the 3D case, we apply our CGC to spatial convolutions in P3D-A or 2D convolutions in TSN or TSM; the pooling layer produces $c \times k \times k \times k$ cubes, the Context Encoding Module encodes $k \times k \times k$ feature maps into a vector of length $k^3/2$ and the Gate Decoding Module generates $o \times c \times t \times k \times k$ gates. Note that for the first convolutional layer, we use $\mathsf{l} \in \mathbb{R}^{3 \times 64}$ for the Channel Interacting Module.

Experiment Setting. The Something-Something (v1) dataset has a training split of 86,017 videos and a validation split of 11,522 videos, with 174 categories. We follow (Qiao et al., 2019) to train on the training set and report evaluation results on the validation set. We follow Lin et al. (2018) to process videos and augment data. Since we only use ImageNet for pretraining, we adapt the code base of TSM but the training setting from Qiao et al. (2019). We train TSN- and TSM-based models for 45 epochs (50 for P3D-A), start from a learning rate of 0.025 (0.01 for P3D-A), and decrease it by 0.1 at 26 and 36 epochs (30, 40, 45 for P3D-A). For TSN- and TSM-based models, the batch size is 64 for 8-frame models and 32 for 16-frame models, and the dropout rate is set to 0.5. P3D-A takes 32 continuously sampled frames as input and the batch size is 64, and the dropout ratio is 0.8. We use the evaluation setting of Lin et al. (2018) for TSN- and TSM-based models and the evaluation settings of Wang et al. (2017b) for P3D-A. All the models are trained with 8-GPU machines.

Performance Comparisons. As Table 3 shows, our CGC significantly improves the performance of baseline CNN models, compared to Non-local (Wang et al., 2017b). As aforementioned, Non-local modules modify the input feature maps of convolutional layers by reassembling local features according to the global correspondence. We apply Non-local blocks in the most effective way as is reported by Wang et al. (2017b). However, we observe that its performance gain is not consistent when training the model from scratch, which again indicates that modifying features according to

Table 4: Machine translation results on IWSLT'14 De-En. Param indicates the number of parameters in the model. Bold indicates the best result.

Models	Param	BLEU-4
(Deng et al., 2018)	-	33.08
Transformer (Vaswani et al., 2017)	39.47M	34.41
DynamicConv (Wu et al., 2019)	38.69M	35.16
LightConv (Wu et al., 2019)	38.14M	34.84
LightConv + Dynamic Encoder (Wu et al., 2019)	38.44M	35.03
LightConv + CGC Encoder (Ours)	**38.15M**	**35.21**

the global correspondence may be misleading in the early training process. When applied to TSM, it even degrades the performance. Our proposed CGC consistently improves the performance of all the baseline models. When applied to TSM, our CGC yields the state-of-the-art performance, when without Kinetics (Carreira & Zisserman, 2017) pretraining, with only RGB modality and with negligible parameter increment.

3.4 MACHINE TRANSLATION

Baseline Methods. The LightConv proposed by Wu et al. (2019) achieves better performances with a lightweight convolutional model, compared to Transformer (Vaswani et al., 2017). We take it as the baseline model and augment their Lightweight Convolution with our CGC. Note that the Lightweight Convolution is a grouped convolution $\mathbf{L} \in \mathbb{R}^{H \times k}$ with weight sharing, so we remove the Channel Interacting Module since we do not need it to project latent representations. We resize the input sequence $\boldsymbol{S} \in \mathbb{R}^{c \times L}$ to $\mathbb{R}^{H \times 3k}$ with average pooling. For those sequences shorter than $3k$, we pad them with zeros. Since the decoder decodes translated words one by one at the inference time, it is unclear how to define global context for it. Therefore, we only replace the convolutions in the encoder.

Experiment Setting. We follow Wu et al. (2019) to train all the compared models with 160K sentence pairs and 10K joint BPE vocabulary. We use the training protocol of DynamicConv (Wu et al., 2019) provided in Ott et al. (2019). The widely-used BLEU-4 (Papineni et al., 2002) is reported for evaluation of all the models. We find that it is necessary to set beam width to 6 to reproduce the results of DynamicConv reported in (Wu et al., 2019), and we fix it to be 6 for all the models.

Performance Comparisons. As Table 4 shows, replacing Lightweight Convolutions in the encoder of LightConv with our CGC significantly outperforms LightConv and LightConv + Dynamic Encoder by 0.37 and 0.18 BLEU, and yields the state-of-the-art performance. As is discussed previously, Dynamic Convolution leverages a linear layer to generate the convolution kernel according to the input segment, which lacks the awareness of global context. This flaw may lead to sub-optimal encoding of the source sentence and thus the unsatisfied decoded sentence. However, our CGC incorporates global context of the source sentence and helps significantly improve the quality of the translated sentence. Moreover, our CGC is much more efficient than Dynamic Convolution because of our module design. Our CGC only needs 0.01M extra parameters, but Dynamic Convolution needs $30\times$ more.

4 RELATED WORKS

There has been much effort in augmenting CNNs with context information. They can be roughly categorized into three types: first, adding backward connections in CNNs (Stollenga et al., 2014; Zamir et al., 2017; Yang et al., 2018) to model the top-down influence (Gilbert & Li, 2013) like humans' visual processing system; second, modifying intermediate feature representations in CNNs according to attention mechanism (Vaswani et al., 2017; Wang et al., 2017b); third, dynamically generating the parameters of convolution according to local or global information (Jia et al., 2016; Noh et al., 2016; Dai et al., 2017; Wu et al., 2019).

For the first category of works, it is still unclear how the feedback mechanism can be effectively and efficiently modeled in CNNs. For example, Yang et al. (2018) proposed an Alternately Updated Clique to introduce feedback mechanisms into CNNs. However, compared to standard CNNs, the complex updating strategy increases the difficulty for training them as well as the latency at the inference time. The second category of works is the global feature interaction methods. They(Vaswani et al., 2017; Wang et al., 2017a;b; Park et al., 2018; Hu et al., 2018; Chen et al., 2019; Cao et al., 2019; Bello et al., 2019) were proposed recently to modify local features according to global context information, usually by a global correspondence, i.e. the self-attention mechanism. There are also works on reducing the complexity of self-attention mechanism (Parmar et al., 2018; Child et al., 2019). However, this family of works only considers changing the input feature maps. The third type of works is more related to our work. As is discussed before, our approach is distinct from them in two key factors.

5 CONCLUSION

In this paper, motivated by neuroscience research on neurons as "*adaptive processors*", we proposed Context-Gated Convolution (CGC) to incorporate global context information into CNNs. Different from previous works which usually modifies input feature maps, our CGC directly modulates convolution kernels under the guidance of global context information. We proposed three modules to efficiently generate a gate to modify the kernel. As such, our CGC is able to extract representative local patterns according to global context. The extensive experiment results show consistent performance improvements on various tasks. There are still a lot of future works that can be done. For example, ew could design task-specific gating modules to fully uncover the potential of the proposed CGC.

REFERENCES

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. *arXiv preprint arXiv:1904.09925*, 2019.

Yue Cao, Jiarui Xu, Stephen Lin, Fangyun Wei, and Han Hu. Gcnet: Non-local networks meet squeeze-excitation networks and beyond. *arXiv preprint arXiv:1904.11492*, 2019.

Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. Report on the 11th iwslt evaluation campaign, iwslt 2014.

Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 433–442, 2019.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 764–773, 2017.

Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pp. 9712–9724, 2018.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1243–1252. JMLR. org, 2017.

Charles D Gilbert and Wu Li. Top-down influences on visual processing. *Nature Reviews Neuroscience*, 14(5):350, 2013.

R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pp. 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL http://proceedings.mlr.press/v9/glorot10a.html.

Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec 2015. doi: 10.1109/iccv.2015.123. URL http://dx.doi.org/10.1109/ICCV.2015.123.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pp. 630–645. Springer, 2016b.

Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7132–7141, 2018.

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

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, 2015.

Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc V Gool. Dynamic filter networks. In *Advances in Neural Information Processing Systems*, pp. 667–675, 2016.

Younghyun Jo, Seoung Wug Oh, Jaeyeon Kang, and Seon Joo Kim. Deep video super-resolution network using dynamic upsampling filters without explicit motion compensation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3224–3232, 2018.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Alex Krizhevsky et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

Ji Lin, Chuang Gan, and Song Han. Temporal shift module for efficient video understanding. *arXiv preprint arXiv:1811.08383*, 2018.

Ben Mildenhall, Jonathan T Barron, Jiawen Chen, Dillon Sharlet, Ren Ng, and Robert Carroll. Burst denoising with kernel prediction networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2502–2510, 2018.

Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814, 2010.

Hyeonwoo Noh, Paul Hongsuck Seo, and Bohyung Han. Image question answering using convolutional neural network with dynamic parameter prediction. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 30–38, 2016.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 311–318. Association for Computational Linguistics, 2002.

Jongchan Park, Sanghyun Woo, Joon-Young Lee, and In So Kweon. Bam: Bottleneck attention module. *arXiv preprint arXiv:1807.06514*, 2018.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.

Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Weight standardization. *arXiv preprint arXiv:1903.10520*, 2019.

Zhaofan Qiu, Ting Yao, and Tao Mei. Learning spatio-temporal representation with pseudo-3d residual networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5534–5542. IEEE, 2017.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015.

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In *Advances in Neural Information Processing Systems*, pp. 2483–2493, 2018.

Marijn F Stollenga, Jonathan Masci, Faustino Gomez, and Jürgen Schmidhuber. Deep networks with internal selective attention through feedback connections. In *Advances in neural information processing systems*, pp. 3545–3553, 2014.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pp. 5998–6008, 2017.

Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaoou Tang. Residual attention network for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164, 2017a.

L. Wang, Y. Xiong, Z. Wang, Y. Qiao, D. Lin, X. Tang, and L. Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 10, 2017b.

Felix Wu, Angela Fan, Alexei Baevski, Yann N Dauphin, and Michael Auli. Pay less attention with lightweight and dynamic convolutions. *arXiv preprint arXiv:1901.10430*, 2019.

Yibo Yang, Zhisheng Zhong, Tiancheng Shen, and Zhouchen Lin. Convolutional neural networks with alternately updated clique. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2413–2422, 2018.

Amir R Zamir, Te-Lin Wu, Lin Sun, William B Shen, Bertram E Shi, Jitendra Malik, and Silvio Savarese. Feedback networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1308–1317, 2017.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pp. 649–657, 2015.

Bolei Zhou, Alex Andonian, Aude Oliva, and Antonio Torralba. Temporal relational reasoning in videos. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 803–818, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

Mohammadreza Zolfaghari, Kamaljeet Singh, and Thomas Brox. Eco: Efficient convolutional network for online video understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 695–712, 2018.

A APPENDIX

A.1 DETAILS OF TRAINING SETTINGS ON IMAGENET AND CIFAR-10

For ImageNet, we use 224×224 random resized cropping and random horizontal flipping for data augmentation. Then we standardize the data with mean and variance per channel. We use a standard cross-entropy loss to train all the networks with a batch size of 256 on 8 GPUs by SGD with a weight decay of 0.0001 and a momentum of 0.9 for 100 epochs. We start from a learning rate of 0.1 and decrease it by a factor of 10 every 30 epochs. For CIFAR-10, we use 32×32 random cropping with a padding of 4 and random horizontal flipping. We use a batch size of 128 and train on 1 GPU. We decrease the learning rate at the 81st and 122nd epochs, and ends training after 164 epochs.

A.2 DETAILS ABOUT P3D-A

Based on ResNet-50, we add a temporal convolution with $k = 5, stride = 2$ after the first convolutional layer. For convolutional layers in residual blocks, we follow Wang et al. (2017b) to add $3 \times 1 \times 1$ convolution (stride is 1) after every two $1 \times 3 \times 3$ convolutions. We only inflate the max pooling layer after the first convolutional layer with a temporal kernel size of 3 and a stride of 2 without adding any other temporal pooling layers. Note that all the aforementioned convolutional layers come with a Batch Normalization layer and a ReLU activation function.