Efficient Representation Learning via Adaptive Context Pooling Supplementary Material

Chen Huang [1] **Walter Talbott** [1] **Navdeep Jaitly** [1] **Josh Susskind** [1]

1. ContextPool in ConvNets

We show our *ContextPool* module can be easily applied to convolutional neural networks. A classical ConvNet is composed of alternating layers of convolution and pooling. After convolution at each layer (often followed by some activation function), assume we have a feature map $\boldsymbol{X} \in \mathbb{R}^{h\times w\times c}$ where h, w, c are the height, width, and the number of channels. For a spatial location (i, j) on the feature map $\boldsymbol{X}$, we use $\boldsymbol{x}_{i,j}$ to denote the corresponding feature vector at that location. The feature map $\boldsymbol{X}$ is then passed to the pooling layer, which aggregates the contextual information within a set of local regions R, producing a pooled feature map $\boldsymbol{Y}$ of smaller size. For the pooling function, common options include average pooling $f_{ave}()$ and max pooling $f_{max}()$. For example, we can have average pooled features $\boldsymbol{y}_k$ as:

$$\boldsymbol{y}_k = f_{ave}(\boldsymbol{X}|R_k) = \frac{1}{|R_k|}\sum_{(i,j)\in R_k} \boldsymbol{x}_{i,j}, \qquad (1)$$

where R_k is the pooling region k in feature map $\boldsymbol{X}$.

There are two main drawbacks with the standard average pooling function: 1) The pooling region R_k is predefined (*e.g.*, 3×3), thus the receptive field remains fixed for each location. However, this is undesirable to encode the contexts or semantics over spatial locations because different locations may correspond to objects with varying scales. 2) The pooing function pays equal attention to all positions in a receptive field, which is usually not the case (Luo et al., 2016). Our ContextPool method addresses these drawbacks by using *learned* pooling weights and support size for each location, aiming to capture meaningful context with varying scale.

Specifically, we learn the normalized maps of pooling weights $W \in \mathbb{R}^{h\times w}$ and pooling sizes $S \in \mathbb{R}^{h\times w}$ together for all positions (see Fig. 1). Both maps are conditioned on the input feature map $\boldsymbol{X}$, *i.e.*, $\{W, S\} = m(\boldsymbol{X})$ with the same spatial resolution with $\boldsymbol{X}$. Note the pooling weights W are normalized by a softmax function in order to apply effective weighting over different positions during pooling. While we learn normalized pooling size $S_{i,j} \in [0, 1]$ mainly to make its learning invariant to feature map size. This way, during the actual pooling for position (i, j), we can easily transform $S_{i,j}$ to the standard deviation $\sigma_{i,j} = r \cdot S_{i,j} \cdot (w + h)/2$ of a Gaussian mask $G \sim \mathcal{N}(i, j, \sigma^2_{i,j}, \sigma^2_{i,j})$. Here r is an empirically set scalar (say 0.05), and $G \in \mathbb{R}^{h\times w}$ imposes spatial locality for pooling.

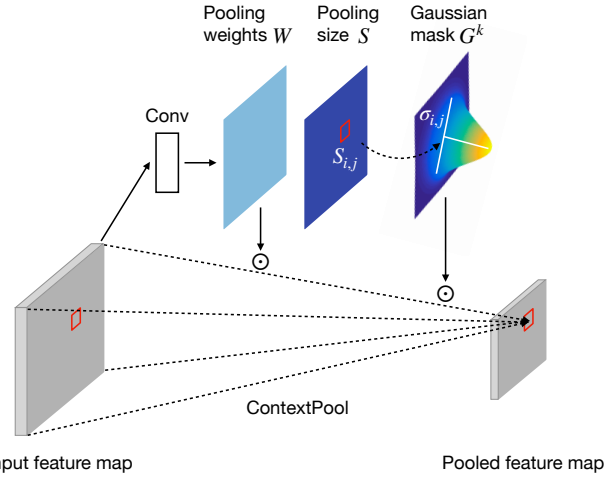

Figure 1. Illustration of our ContextPool module in ConvNets.

Finally, given the pooling weights W and Gaussian mask G^k for the pooling center k, our ContextPool module aggregates information across all the spatial positions in input feature map $\boldsymbol{X}$. In other words, ContextPool operates on the 2D spatial domain for the 3D input $\boldsymbol{X}$, and the operation remains the same across the channel dimension:

$$\boldsymbol{y}_k = f_{ave}(\boldsymbol{X} \odot \gamma(W) \odot \gamma(G^k)) = \sum_{i,j} \boldsymbol{x}_{i,j} \cdot W_{i,j} \cdot G^k_{i,j}, \qquad (2)$$

where $\gamma(\cdot)$ is a broadcasting function to accommodate element-wise multiplication $\odot$. The normalization factor is set as $C(\boldsymbol{X}) = \sum_{i,j} W_{i,j} \cdot G^k_{i,j}$.

[1]Apple Inc., Cupertino, United States. Correspondence to: Chen Huang <chen-huang@apple.com>.

Proceedings of the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, PMLR 162, 2022.

In practice, the prediction function $m(\cdot)$ for pooling weights W and sizes S is implemented by applying two convolutional layers over the feature map $\boldsymbol{X}$. During training, the convolutional kernels for both the main network and ContextPool are learned simultaneously. We show our ContextPool is pretty lightweight with small increase in model size, and is able to consistently improve performance. We validate this on two common benchmarks for image classification, as we now demonstrate.

2. Results on Image Classification

CIFAR-10 dataset We first evaluate ConvNets equipped with ContextPool (CP) for image classification on CIFAR-10 dataset (Krizhevsky, 2009). CIFAR-10 consists of 60k images with 10 classes. We follow the standard training and testing protocol, using 50k images for training a ResNet (He et al., 2016) and 10k images for testing.

Table 1 shows the ResNet-44 baseline with regular pooling function obtains 92.9% accuracy on CIFAR-10. While DCN and N-Jet-based methods are parameter-efficient when learning adaptive kernel size using Gaussian derivative filters. They show success of learning data-dependent receptive fields, but the performances are not as competitive as those of other methods. Note the results are from the original papers using only small model sizes. It remains unclear how performance scales with increasing model size. On the other hand, deformable ConvNets (Dai et al., 2017) learn spatial offsets for the sampling locations of convolution and pooling operations, offering an alternative way for learning adaptive receptive field. We observe that both the deformable convolution and deformable pooling modules contribute to compelling results.

In comparison, our CP-improved ResNets achieve a better trade-off between performance and parameter efficiency than deformable ConvNets. When applied to the same ResNet-44 backbone, our CP already achieves a competitive accuracy of 93.4% at low overhead. We can further improve accuracy to 93.7% by training a deeper network with CP. Note the resulting CP-ResNet-46 outperforms deformable ConvNets with a similar model size.

Lastly, we offer two more variants of ContextPool in the ConvNet framework. For the first variant, we only learn adaptive pooling size, with uniform pooling weights (*i.e.*, average pooling). This baseline is analogous to those learning methods for pooling region or receptive field (Coates & Ng, 2011). Another related method is spatial pyramid pooling (He et al., 2014). But this method is not directly comparable because it is mainly designed to deal with input images of varying size. Table 1 (bottom cell) shows that our pooling size learning performs slightly worse than deformable pooling (Dai et al., 2017). More importantly, it is inferior to our full method due to the lack of dynamic pooling weights. When we replace our learned pooling weights with those defined by the feature similarity (as done for transformers in main paper), we see marginal improvements which indicates the need of pooling weights learning.

Table 1. Model size and performance (%) on CIFAR-10. Results are reported over three runs per setting.

Method	Size	Accuracy
ResNet-44 (He et al., 2016)	0.66M	92.9
DCN (Tomen et al., 2021)	0.47M	89.7±0.3
N-Jet-ResNet-32 (Pintea et al., 2021)	0.52M	92.3±0.3
Deform ResNet-44 (Pool) (Dai et al., 2017)	0.68M	93.2±0.4
Deform ResNet-44 (Pool+Conv) (Dai et al., 2017)	0.69M	93.5±0.2
CP-ResNet-44	0.68M	93.4±0.3
CP-ResNet-46	0.70M	**93.7±0.2**
CP-ResNet-44 (learn pooling size only)	0.67M	93.1±0.2
CP-ResNet-44 (pooling weights by fea similarity)	0.67M	93.2±0.2

Table 2. Classification accuracy (%) and model size on ImageNet.

Backbone	Method	Top-1	Top-5	Size
ResNet-50	baseline	76.5	93.1	26.6M
	Deform (Dai et al., 2017)	76.6	93.2	26.8M
	CP-baseline	**77.3**	**93.6**	26.8M
ResNet-101	baseline	78.4	94.2	45.5M
	Deform (Dai et al., 2017)	78.4	94.2	45.8M
	CP-baseline	**78.9**	**94.4**	45.8M

ImageNet-1K dataset We further compare our CP-improved ResNets with the strong baseline of deformable ConvNets (Dai et al., 2017) on ImageNet-1K dataset. For a fair comparison, we use the same training and inference settings as in (Dai et al., 2017). Table 2 illustrates the validation-set results based on two ResNet backbones. It can be observed that our CP-ResNets achieve consistent improvements over both the baseline and deformable ConvNets, without large increase in model size. Our hypothesis is that CP benefits more from its strong context modeling capability on high-resolution ImageNet images. For future work, it would be interesting to test our approach on various image resolutions or on more types of tasks that have different needs for a context model.

References

Coates, A. and Ng, A. Selecting receptive fields in deep networks. In *NeurIPS*, 2011.

Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., and Wei, Y. Deformable convolutional networks. In *ICCV*, 2017.

He, K., Zhang, X., Ren, S., and Sun, J. Spatial pyramid pool-

ing in deep convolutional networks for visual recognition. In *ECCV*, 2014.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Krizhevsky, A. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Luo, W., Li, Y., Urtasun, R., and Zemel, R. Understanding the effective receptive field in deep convolutional neural networks. In *NeurIPS*, 2016.

Pintea, S., Tömen, N., Goes, S., Loog, M., and van Gemert, J. Resolution learning in deep convolutional networks using scale-space theory. *IEEE Transactions on Image Processing*, 30:8342 – 8353, 2021.

Tomen, N., Pintea, S.-L., and Van Gemert, J. Deep continuous networks. In *ICML*, 2021.