Rectified Convolution

Hang Zhang
Amazon
hzaws@amazon.com

Abstract

We propose a simple modification of the convolutional layer by rectifying the feature-map activations, which alleviates the biases introduced via zero-padding. We refer to this modified version of convolution as *rectified convolution*. Comparing to standard convolutional layer, the rectified version almost introduces no extra computation cost or memory usage. Simply replacing the convolutional layer with the rectified version, the performances of CNNs are consistently improved, for example the top-1 accuracy of ResNet-50 on ImageNet is increased from 76.5% to 77.1%.

1 Introduction

Convolutional neural network (CNN) has become the pre-dominant method in computer vision. CNN learns the feature representation directly from the data by stacking the convolutional layers with non-linearities and downsampling. There are a lot of research study on non-linearity functions and regularization methods, but the convolutional layer has stayed almost unchanged since LeNet [4]. In this paper, we rethinking the boundary padding for convolutional layers.

Modern CNN architectures often follow a modular design which stacks multiple blocks in the same type at each stage [7, 2]. To preserve the same spatial resolution at each stage, zero-padding is used for convolutional layers. However, the input feature-map is often ReLU activated with a non-zero mean. As convolutional layers act in a sliding window manner, the resulting activations along the feature-map boundaries have padded zeros as the input, which makes it hard to estimate the batch statistics reliably using Batch Normalization. In addition, the ratio between the boundary pixels and the interior pixels change with the input size, which may lead to discrepancy in the feature representation using different training and validation image resolutions [9].

For the image-to-image problem, the generative model often employs reflection-padding to alleviates the artifacts along the boundaries due to the padding [11]. Despite its success in generative model, the reflection-padding is not suitable for image classification network due to introducing aliasing artifacts. A question naturally arises *what is the idea padding value along the feature-map boundaries*? Dropout [8] randomly masks out ratio p of the activations in the feature-map, and scales up the remaining activations by $\frac{1}{1-p}$ during the training. And it serves as an identity layer during the inference to form a in-network ensemble of various modes during the training. Inspired by the dropout work, we treat the padded zeros as the missing pixels just like being masked out along the boundaries.

As the main contribution of this paper, we introduce a *rectified convolution*, where the boundary values are rectified to alleviates the biases introduced by zero-padding. We simply scale up the activation by a ratio of $\frac{kernel\_size}{valid\_pixels}$ to compensate the padded zeros, so that the expectation of the resulting feature-map mean is consistent across spatial locations. Experimental results have shown that the proposed rectified convolution can improve the classification accuracy on ImageNet [1] consistently for different CNN architectures. We have also studied the behavior of using average aggregation mode instead of summation in the convolutional layers, and have achieved similar performance. It

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

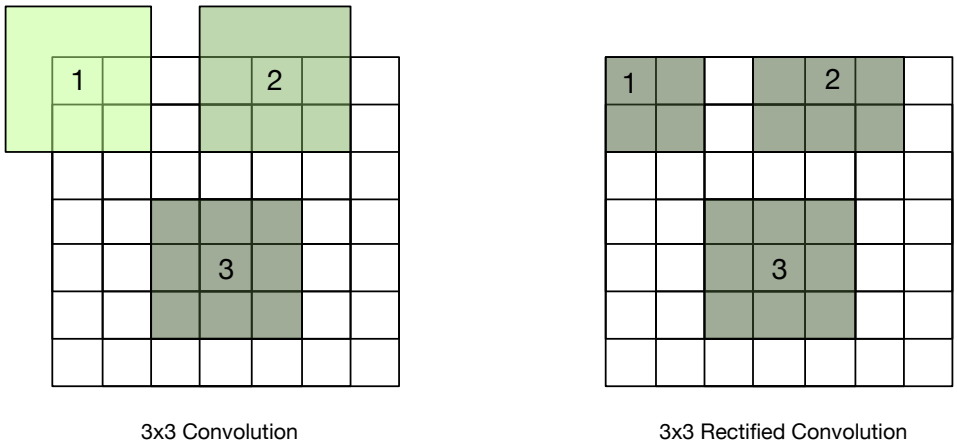

Figure 1: Comparing to the standard 2D convolution with zero-padding, rectified convolution only takes the valid input values and scale up the activations based on number of valid input elements. The kernel size is 3×3 and feature-map size is 7×7.

demonstrates that the discrepancies are mainly caused by unbalanced valid pixel ratio instead of aggregation mechanism. This modification is simple and can be easily implemented as an in-place operator using deep learning toolkit [1], with almost no extra computation cost or memory usage.

2 Methods

In this section, we use 2D convolution as an example to describe how rectified convolution works. The convolutional layer can be regarded as a 2D convolution if it has stride of 1 and a single input and output channel.

2.1 2D Convolution with Zero-padding

For a zero-padded 2D convolution with an input feature-map $x \in \mathbb{R}^{H \times W}$ and kernel $k \in \mathbb{R}^{m \times n}$, the value of the output feature-map $y \in \mathbb{R}^{H \times W}$ at the location $[h, w]$ can be represented as:

$$y[h, w] = \sum_{i=1}^{m} \sum_{j=1}^{n} k[i, j] \cdot \hat{x}[h - i, w - j], \tag{1}$$

where $\hat{x}[h, w]$ is the input pixel value at the location $[h, w]$ or padded zero:

$$\hat{x}[h, w] = \begin{cases} x[h, w] & \text{if } \delta(x[h, w]) \\ 0, & \text{otherwise,} \end{cases} \tag{2}$$

where $\delta()$ is an indicator function which indicates whether the pixel is valid (inside the input feature-map):

$$\delta(x[h, w]) = (0 \leq h \leq H) \wedge (0 \leq w \leq W). \tag{3}$$

Boundary Effects and Feature Discrepancy. The convolution operation acts in a sliding window manner. The number of valid input pixels inside the convolutional kernel depends on the pixel location of the input, as shown in Figure 1. The input feature-map of convolutional layer in modern CNN is usually ReLU activated and has a non-zero mean. The activations along the feature-map boundaries are biased due to zero padding. In addition, the ratio of non-biased activations varies with

[1] PyTorch implementation will be released upon publication.

Network	Rectified	acc%
ResNet-50 [2]		76.48
	✓	77.10 (+0.62)
ResNet-101 [2]		78.13
	✓	78.74 (+0.41)
ResNeXt-50 [10]		78.17
	✓	78.48 (+0.31)
RegNetX-4GF [6]		79.03
	✓	79.35 (+0.33)
ResNeSt-50† [12]		78.73
	✓	79.38 (+0.65)

Table 1: Image classification results on ImageNet. Rectified convolution consistently improves the performance for different CNN architectures. (ResNeSt-50† is using $2s8x$ fast setting with 4.5GFLOPs and 26.4M params.)

different input sizes. For example 36 pixels are non-biased for input feature-map size of 7×7 using a 3×3 convolution kernel and resulting in a valid ratio of $36/49 = 0.735$, while the valid ratio for 6×6 is 0.694.

Batch normalization [3] is often used as a regularization after the convolutional layer in CNNs. The accumulation of batch statistics becomes difficult to establish due to the biased activations along the boundaries. Furthermore, the estimated batch statistics is conditional on the input resolution change, leading to the discrepancy in the learned network representation.

2.2 Rectified Convolution

To tackle the difficulty of inconsistent number of valid input pixels along the feature-map boundaries, we simply scale up the resulting activation:

$$y[h, w] = \frac{mn}{v[h, w]} \sum_{i=1}^{m} \sum_{j=1}^{n} k[i, j] \cdot \hat{x}[h - i, w - j], \tag{4}$$

where $v[h, w]$ is the number of valid pixels residing inside the feature-map for the pixel location at h, w:

$$v[h, w] = \sum_{i=1}^{m} \sum_{j=1}^{n} \mathbf{1}[(0 \leq h - i \leq H) \wedge (0 \leq w - j \leq W)]. \tag{5}$$

3 Experimental Results

In this section, we first compare the image classification performance if different CNN architectures [2, 10, 12, 6] using rectified convolution and standard 2D convolution with zero-padding on ImageNet dataset [1], then study the different aggregation mode in the rectified convolution.

Implementation Detail For easily measure the improvement of rectified convolution, we train different network architectures in the same setting. We use data parallel training on 8 GPUs with data sharding, where the mini-batch on each GPU is sampled from the corresponding shard without replacement. For data augmentation, only standard transformations are used, including random size cropping between 0.08 to 1.0 of the original image area, random horizontal flip, color jittering and lighting changes. After jittering, the image is subtracted by RGB mean and divided by the standard deviation before fed into the network. Batch Normalization [3] and ReLU activation function [5] are used after each convolutional layer. A mini-batch size of 512 is used with 64 image samples per GPU. We use a learning rate of 0.2 and a weight decay of 0.0001. No regularization method is used except for weight decay. The networks are trained on the ImageNet training set for 120 epochs with a cosine learning rate decay and evaluated on the validation set.

Network	Rectified	acc%
ResNet-50 [2]	sum	77.10
	avg	77.11

Table 2: Sum aggregation v.s. average aggregation. Image classification results on ImageNet.

Rectified Convolution We evaluate the performance of difference CNN models using rectified convolution or standard convolution with zero-padding, including ResNet [2], ResNeXt [10], ResNeSt [12] and RegNet [6]. The image classification top-1 accuracy on ImageNet [1] validation set is reported in Table 1. We can see that the performances of these networks are boosted by $0.3 - 0.6\%$ on top-1 accuracy.

Sum mode v.s. Average Mode Since rectified convolution mainly balances the activation scale regardless of the number of valid input elements, it is naturally to consider using average mode to aggregate the convolution instead of summation. In this way, it is the same as averaging the responses only on the valid input elements:

$$y[h, w] = \frac{1}{v[h, w]} \sum_{i=1}^{m} \sum_{j=1}^{n} k[i, j] \cdot \hat{x}[h - i, w - j], \tag{6}$$

$v[h, w]$ is the count of valid input at output location h, w as in Equation 4. We compare the performance of ResNet-50 on ImageNet, and the results are shown in Table 2. We get very similar performance using different aggregation mode, which demonstrates that the biases are main introduced by inconsistent valid input elements between boundary and interior pixels instead of aggregation mode.

4 Conclusion

In this paper, we propose a simple modification of standard convolutional layer with zero-padding by rectifying the activations with padded zero input along the boundaries. We refer to this modified version as rectified convolution. The modification is simple yet efficient, and consistently improves the performance of CNNs without introducing extra computation or memory usage. As the rectified convolution alleviates the aliasing along the feature-map boundaries, we expect it could be beneficial to generative models or segmentation CNNs.

References

[1] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR09 (2009)

[2] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. arXiv preprint arXiv:1512.03385 (2015)

[3] Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: International Conference on Machine Learning. pp. 448–456 (2015)

[4] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE **86**(11), 2278–2324 (1998)

[5] Nair, V., Hinton, G.E.: Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10). pp. 807–814 (2010)

[6] Radosavovic, I., Kosaraju, R.P., Girshick, R., He, K., Dollár, P.: Designing network design spaces. arXiv preprint arXiv:2003.13678 (2020)

[7] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[8] Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of machine learning research **15**(1), 1929–1958 (2014)

[9] Touvron, H., Vedaldi, A., Douze, M., Jégou, H.: Fixing the train-test resolution discrepancy. In: Advances in Neural Information Processing Systems. pp. 8250–8260 (2019)

[10] Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. arXiv preprint arXiv:1611.05431 (2016)

[11] Zhang, H., Dana, K.: Multi-style generative network for real-time transfer. arXiv preprint arXiv:1703.06953 (2017)

[12] Zhang, H., Wu, C., Zhang, Z., Zhu, Y., Zhang, Z., Lin, H., Sun, Y., He, T., Muller, J., Manmatha, R., Li, M., Smola, A.: Resnest: Split-attention networks. arXiv preprint arXiv:2004.08955 (2020)