DISTRIBUTION PADDING IN CONVOLUTIONAL NEURAL NETWORKS

Anh-Duc Nguyen, Seonghwa Choi, Woojae Kim, Sewoong Ahn, Jinwoo Kim, and Sanghoon Lee

Yonsei University

ABSTRACT

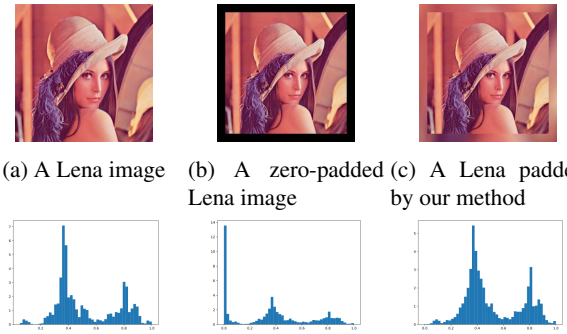
Even though zero padding is usually a staple in convolutional neural networks to maintain the output size, it is highly suspicious because it significantly alters the input distribution around border region. To mitigate this problem, in this paper, we propose a new padding technique termed as distribution padding. The goal of the method is to approximately maintain the statistics of the input border regions. We introduce two different ways to achieve our goal. In both approaches, the padded values are derived from the means of the border patches, but those values are handled in a different way in each variant. Through extensive experiments on image classification and style transfer using different architectures, we demonstrate that the proposed padding technique consistently outperforms the default zero padding, and hence can be a potential candidate for its replacement.

Index Terms— Deep learning, convolutional neural network, image padding

1. INTRODUCTION

Convolutional neural networks (CNNs) have played an important role in the advancement of most areas in image processing and computer vision in recent years [1–8]. Because of the large performance gap between deep learning and traditional models, it is not surprising that huge effort has been invested in tweaking the mechanisms of CNNs to constantly improve the state of the art. Most studies concentrate on architectures [9,10], initialization [11,12], optimization [13,14], activation [12], and so on. However, little attention has been paid to padding, which is easily the most essential part of the convolution operation. The main purpose of padding is to maintain the border information and keep the input resolution fixed. Otherwise, the output becomes smaller after each convolution and the spatial information at the borders is washed away quickly [15].

Since zero padding is simple and works well in many tasks, it is perhaps the most common technique in practice. It works by taping zeroes, which are irrelevant data, along the border of the input. This, however, seriously alters the spatial distribution of the image at its borders. It may seem that the ratio between the border regions and whole image is extremely small, which may have little impact on the performance of the CNNs. We argue that while the former is true, the latter may not because the filters are shared spatially, and hence they have to adapt themselves to the distributions of both the plausible pixel values as well as the spooky ones at the borders, which might hinder the learning of the networks. Even if the input has zero mean, zero padding shrinks the variances, which still confuses the CNNs. This can be seen as a form of an implicit and unwanted regularization on the filters. Particularly, it is reminiscent of Dropout [16] but without any randomness. An example of a zero-padded image and its histogram of the top region corresponding to the top stride of the convolutional filters are shown in Figs. 1(b)&(e). Clearly, the method seriously changes the distribution of the border region by introducing a new mode at 0.



(a) A Lena image (b) A zero-padded Lena image (c) A Lena padded by our method

(d) A histogram of the top region of (a) (e) A histogram of the top region of (b) (f) A histogram of the top region of (c)

Fig. 1: (a) A Lena image padded by (b) zero padding, (c) our method (mean-interpolation padding), and their corresponding histograms (d-f). Best viewed in color and zoom.

There has been several alternatives to zero padding. Firstly, padding with constant other than zero is also a legit choice although it is hardly selected in practice. Also, circular padding aims to simulate circular convolution, but its use is only in some specific areas [17]. Among all the options, reflection padding is the most popular scheme. It works by reusing the border data but in a reverse direction. At face value, this seems to eliminate the problem of zero padding because the padded values are from the input. How-

This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1702-08.

ever, due to the multiplicity of the data, the distribution is still altered, which in fact does not solve any problem posed by zero padding at all. Recently, an implicit padding method called partial-convolution padding has been proposed [18]. In each stride of the convolution, the method considers only the valid region in the patch, and then rescales based on the ratio between the whole patch and the area of the valid pixels. However, due to its definition, the ratio is always larger than 1, so the output can be larger or smaller than it should be.

In this paper, we offer practitioners another choice by presenting a padding scheme named distribution padding. The method aims to preserve the local spatial distributions of the data at the borders of the input, which justifies the name of the method. An example of the padded image can be seen in Fig. 1(c)&(f). Clearly, the method seems to smooth the distribution of the border but it still keeps the same shape and modes of the original. The padding works by considering the local means of the input at its borders. Specifically, we propose two variants of the method, which differs in the way the means are handled. Through thorough experiments with various models in image classification and style transfer, we demonstrate that our method outperforms zero padding in these tasks and is efficient enough to take over its lead role in convolution.

2. DISTRIBUTION PADDING

In order to maintain the statistics of an image at the borders, one intuitive way is to pad the image with the means of the border regions. However, it is not obvious the mean values of which patches should be used for padding because a padded pixel is usually seen by filters multiple times during convolution. Also, if the receptive fields at the borders are padded with the means of the valid regions, then the variances will be underestimated. In this respect, we propose two variants of our padding scheme that can resolve both the problems.

2.1. Mean-interpolation padding

Since input pixels are usually seen multiple times by the network filters, the padded values should be defined to be compatible with multiple neighboring regions. To this end, we propose to pad the image with the interpolated means of the input patches corresponding to the sliding windows when performing convolution. The rationale is that image pixels are locally correlated, and so the mean values of the nearby patches are similar to each other. Also, padding the input by its interpolated local means directly preserves the first-order statistics in each local region.

Let us denote the input feature map to a convolutional layer as I and its value at a tuple coordinate p is $I(p)$. Suppose the size of the input is $h \times w$ and the filters in the convolutional layer have height h_f and width w_f. To calculate the means, we simply perform a valid convolution between the input and a box filter as

Algorithm 1 Mean-interpolation padding

Input: input I of shape $h \times w$, padding size $(\lfloor h_f/2 \rfloor, \lfloor w_f/2 \rfloor)$, box filter of shape $h_f \times w_f$
Output: $\tilde{I}$

$\hat{I} := convolve\,(I, b)$
$\tilde{I} := resize\left(\hat{I}, (h + 2h_f//2, w + 2w_f//2)\right)$
$\tilde{I}\,[h_f//2 : h + h_f//2, w_f//2 : w + w_f//2] := I$

$$\hat{I} = I * b, \tag{1}$$

where $b(p) = \frac{1}{h_f * w_f} \forall p$ is a box filter of size $h_f \times w_f$ and "$*$" indicates a linear convolution operator. Because the convolution is valid, the size of $\hat{I}$ is $(h - 2\lfloor h_f/2 \rfloor) \times (w - 2\lfloor w_f/2 \rfloor)$ where "$\lfloor \cdot \rfloor$" denotes the floor operator. To maintain the size after the convolutional layer, the input should be of size $(h + 2\lfloor h_f/2 \rfloor) \times (w + 2\lfloor w_f/2 \rfloor)$. Therefore, we simply resize $\hat{I}$ to the desired size by bilinear interpolation to become $\hat{I}'$. This interpolation kills two birds with one stone: (1) it produces the values that are compatible with their surrounding neighbors, and (2) it introduces some necessary correlated "noise" so that the variance in each padded image patch is not severely underestimated. Finally, the center region of $\hat{I}'$ is replaced with the original input I. Specifically, let the output of the padding scheme be $\tilde{I}$ and $\mathcal{S}$ be the set of indices of pixels that are inside the rectangular box in $\tilde{I}$ defined by two vertices $(\lfloor h_f/2 \rfloor, \lfloor w_f/2 \rfloor)$ and $(h + \lfloor h_f/2 \rfloor, w + \lfloor w_f/2 \rfloor)$. $\tilde{I}$ is defined as

$$\tilde{I}(p) = \begin{cases} \hat{I}'(p) & if\ p \notin \mathcal{S} \\ I(p') & otherwise \end{cases}, \tag{2}$$

where $p' = p - (h_f, w_f)$. The pseudocode of the algorithm is shown in Algorithm 1.

2.2. Mean-reflection padding

Reflection padding is carried out by duplicating the image values along the borders but in reverse order. Inspired by the way reflection padding works, we introduce a variant of the above scheme called mean-reflection padding. Specifically, after obtaining $\hat{I}$, we first resize it to the original size $h \times w$ instead of the padded size, and then pad it using reflection padding to obtain $\hat{I}'$ of size $(h + 2\lfloor h_f/2 \rfloor) \times (w + 2\lfloor w_f/2 \rfloor)$. As above, the last step is to replace the center regions of $\hat{I}'$ by the input data. A summary of the technique is shown in Algorithm 2. We further note that in both variants, we DO NOT backprop the error through the padded regions.

3. EXPERIMENTAL RESULT

In this section, we demonstrate how different padding routines work in image classification and image style transfer.

Algorithm 2 Mean-reflection padding

Given: input I of shape $h \times w$, padding size $(\lfloor h_f/2 \rfloor, \lfloor w_f/2 \rfloor)$, box filter of shape $h_f \times w_f$.

Output: $\tilde{I}$

$$\hat{I} := convolve\,(I, b)$$
$$\tilde{I} := resize\left(\hat{I}, (h, w)\right)$$
$$\tilde{I} := reflection\_pad\left(\tilde{I}, (h_f//2, w_f//2)\right)$$
$$\tilde{I}\,[h_f//2 : h + h_f//2, w_f//2 : w + w_f//2] := I$$

All the codes for the experiments are in Python/Theano[1] [19].

3.1. Image classification

3.1.1. Setup

In this task, we trained CNNs on the training set of CIFAR-10 [20], which contains $50,000$ images equally distributed in 10 categories. We slightly resized the images from 32×32 to 48×48 so that the valid convolution can be carried out in the late layers. To verify the benefit of the proposed padding scheme, we enlisted two of the most well-known network architectures: VGG19 [4] and ResNet34 [10]. While VGG19 is a vanilla stack of convolutional layers with 3×3 filters, ResNet34 is made up of residual blocks in which information flow is enhanced by an identity mapping from early layers. We note that for ResNet34, we removed the first pooling layer and for VGG19, we used batch normalization [13] after each convolutional layer and discarded all the fully connected layers as well. We optimized the multinoulli cross-entropy loss between the softmax outputs and ground truth labels using Adam [14]. We considered zero padding, reflection padding, and partial-convolution-based padding [18] as references in our benchmark. For each padding scheme, we ran totally 5 times, each time 100 epochs. We tested the networks on the test set every $1,000$ iterations to plot the classification error rates. Needless to say, except for padding, we kept all settings the same.

3.1.2. Result

The classification errors of each padding method are demonstrated in Fig. 2. As can be seen, in VGG19, zero padding has the worst performance compared to other schemes. On the other hand, mean-interpolation and mean-reflection schemes achieve the best error rates, which reinforces our statement that the invalid distribution of images harms the learning of CNNs. Meanwhile, reflection padding and partial-convolution have a similar performance and both methods are slightly better than zero padding. This observation is consolidated by the numerical results in the first column in Table 1, which demonstrates the average error rates in the five most recent tests. It is obvious that the proposed schemes outperform all the other methods significantly. However, the story is slightly different when it comes to ResNet34. While mean-interpolation still defends its first place, mean-reflection padding goes from the second best to worst. The reason possibly is that the border mean values are repeated by reflection padding, and the genuine border information are added via the residual connection, the pixel distributions along the borders are hugely altered, which leads to the poor performance of the method. Reflection padding and partial-convolution are still competitive, followed by zero padding. This once again is made clearer by the quantitative results in the second column of Table 1.

It is noticeable that the influence of the proposed method on ResNet34 is not as substantial as VGG19. It is because different from the straightforward sequence of convolutional layers in VGG19, the residual connection in a ResNet34 block helps carry plausible border information from early layers and

[1] Available at https://git.io/fh97z

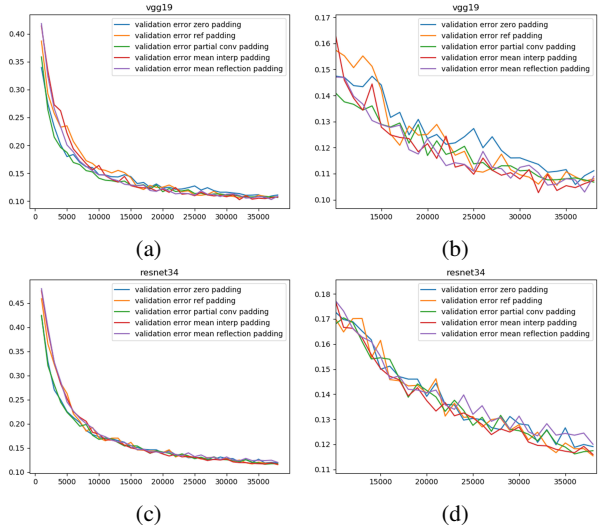


Fig. 2: Classification error rates on test data during training. (a) and (c) are the error rates obtained by VGG19 and ResNet34, respectively. (b) and (d) magnify the last $28,000$ iterations in (a) and (c), respectively. Best viewed in color and zoom.

Table 1: Average classification error rates on test data during the last 5 test times. The best and second best errors obtained by each architecture are in blue and red, respectively.

Padding scheme	VGG19 (%)	ResNet34 (%)
Zero	10.97	12.09
Reflection	10.80	11.78
Partial-convolution	10.77	11.80
Mean-interpolation	10.54	11.74
Mean-reflection	10.67	12.33

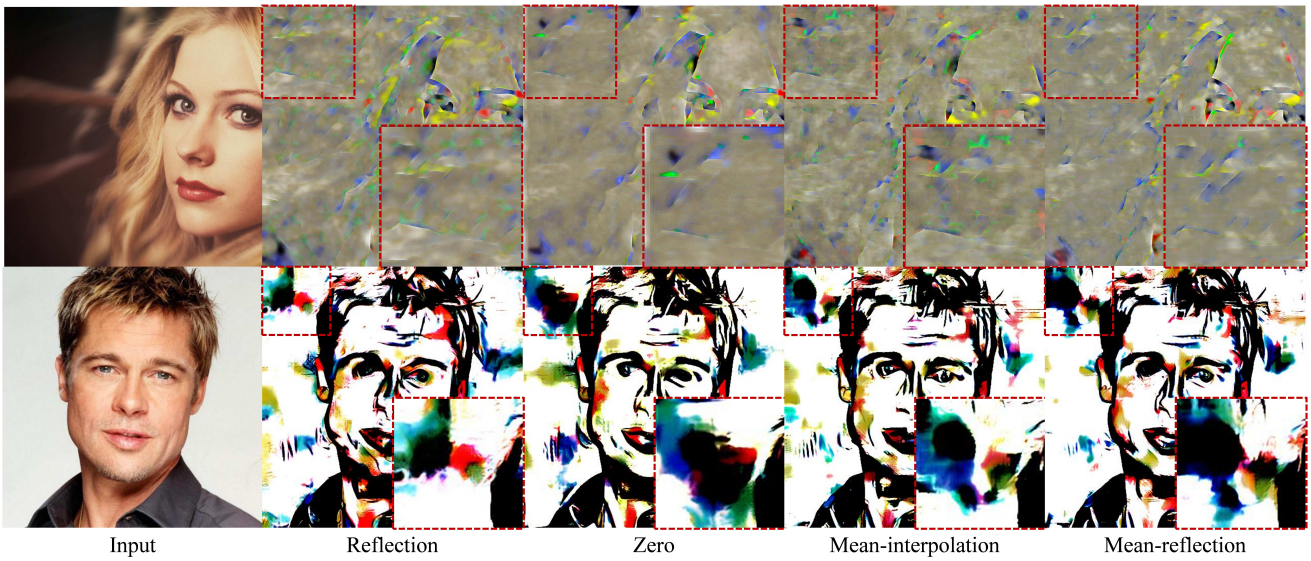


Fig. 3: From left to right: Input images, reflection padding, zero padding, mean-interpolation padding, and mean-reflection padding. Notice the border of the images. Best viewed in color and zoom.

adds it to the output of the block, which dampens the "border effect" in zero padding. This bit of preserved information maintains more or less the distribution along the feature map border, and so the performance is not as severely affected.

In regard to the processing time, mean-interpolation takes roughly the same time with reflection padding in our implementation while mean-reflection takes moderately longer, which is obvious because it includes the reflection padding per se. Zero padding is the fastest method because of not only its simplicity but also its CuDNN implementation. Partial-convolution is second to zero padding in terms of speed. All in all, our method is efficient and practical enough to be an alternative to contemporary schemes such as zero padding and reflection padding.

3.2. Image style transfer

3.2.1. Setup

For the style transfer task, we employed WCTnet [5] as the backbone. The method works by hooking up five different autoencoding models together and the transfer of style is carried out by a whitening and coloring transform in the latent space of each network. The encoders are different parts of a pre-trained VGG19 and the decoders are basically symmetric to the corresponding encoders. To benchmark a padding scheme, we employed it in both the VGG encoder and the decoders. We trained the decoders on the MS COCO training split [21] for 3 epochs using the default hyperparameter choices. The testing was carried out on a separate set provided in the original paper. A complete list of settings can be found in the original paper [5]. For references, we displayed only the results by zero padding and reflection padding because we failed to train the largest decoder using partial-convolution padding with the default hyperparameters.

3.2.2. Result

The visual results of different padding methods are exhibited in Fig. 3. This task requires the use of VGG-like architectures because other architectures perform poorly in this task [22]. Hence, the weakness of zero padding is further revealed. It is fairly easy to spot the striping artifacts at the borders in the results by zero padding. On the other hand, the results obtained by the two proposed techniques are artifact-free, and are similar to those of reflection padding, which is the default choice in the original paper, in terms of performance. The failure of partial-convolution may be due to the scaling problem discussed above, and it suggests that the method might not be suitable in some areas.

4. CONCLUSION

In this paper, we have introduced an image padding method to maintain not only the size of the convolution output but also the spatial distribution of the input border data. We presented two variants both of which are based on the first-order statistics of the input border. Thorough experiments showed that our proposed technique not only outperformed the most widely used scheme, zero padding, in image classification and style transfer but also was on par or better than contemporary alternatives in these tasks.

5. REFERENCES

[1] Jongyoo Kim, Anh-Duc Nguyen, Sewoong Ahn, Chong Luo, and Sanghoon Lee, "Multiple level feature-based universal blind image quality assessment model," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. pp. 291–295, IEEE. 1

[2] Woojae Kim, Jongyoo Kim, Sewoong Ahn, Jinwoo Kim, and Sanghoon Lee, "Deep video quality assessor: From spatio-temporal visual sensitivity to a convolutional neural aggregation network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 219–234. 1

[3] Jongyoo Kim, Anh-Duc Nguyen, and Sanghoon Lee, "Deep cnn-based blind image quality predictor," *IEEE transactions on neural networks and learning systems*, , no. 99, pp. 1–14, 2018. 1

[4] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014. 1, 3

[5] Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang, "Universal style transfer via feature transforms," in *Advances in Neural Information Processing Systems*, pp. 386–396. 1, 4

[6] Anh-Duc Nguyen, Seonghwa Choi, Woojae Kim, and Sanghoon Lee, "A simple way of multimodal and arbitrary style transfer," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. pp. 1752–1756, IEEE. 1

[7] Sewoong Ahn and Sanghoon Lee, "Deep blind video quality assessment based on temporal human perception," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. pp. 619–623, IEEE. 1

[8] Jongyoo Kim, Hui Zeng, Deepti Ghadiyaram, Sanghoon Lee, Lei Zhang, and Alan C Bovik, "Deep convolutional neural models for picture-quality prediction: Challenges and solutions to data-driven image quality assessment," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 130–141, 2017. 1

[9] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger, "Densely connected convolutional networks," in *CVPR*, vol. 1, p. 3. 1

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778. 1, 3

[11] Xavier Glorot and Yoshua Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Aistats*, vol. 9, pp. 249–256. 1

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, pp. 1026–1034. 1

[13] Sergey Ioffe and Christian Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015. 1, 3

[14] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. 1, 3

[15] Andrej Karpathy, "Cs231n convolutional neural networks for visual recognition," *Neural networks*, vol. 1, 2016. 1

[16] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014. 1

[17] Tsun-Hsuan Wang, Hung-Jui Huang, Juan-Ting Lin, Chan-Wei Hu, Kuo-Hao Zeng, and Min Sun, "Omnidirectional cnn for visual place recognition and navigation," *arXiv preprint arXiv:1803.04228*, 2018. 1

[18] Guilin Liu, Kevin J. Shih, Ting-Chun Wang, Fitsum A. Reda, Karan Sapra, Zhiding Yu, Andrew Tao, and Bryan Cantazaro, "Partial convolution based padding," Report, 2018. 2, 3

[19] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: A cpu and gpu math compiler in python," in *Proceedings of the 9th Python in Science Conference*, pp. 1–7. 3

[20] Alex Krizhevsky and Geoffrey Hinton, "Learning multiple layers of features from tiny images," Report, Citeseer, 2009. 3

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. pp. 740–755, Springer. 4

[22] Alexander Mordvintsev, Nicola Pezzotti, Ludwig Schubert, and Chris Olah, "Differentiable image parameterizations," *Distill*, vol. 3, no. 7, pp. e12, 2018. 4