Flipping Data Augmentation of Convolutional Neural Networks Using Discrete Cosine Transforms

Izumi Ito
Information and Communications Engineering
Tokyo Institute of Technology
Tokyo, Japan
ito@ict.e.titech.ac.jp

***Abstract*—Convolutional neural networks (CNNs) are widely used in many areas. The problem now is to collect large numbers of labeled images in order to improve network performance. Data augmentation is to increase the number of images for training, where images are artificially generated by transformation, such as rotation, translation, scaling, and flipping. In this paper, we focus on flipping data augmentation and present a novel algorithm of convolution that involves flipping data augmentation in CNNs. Without generating flipped images beforehand, we can obtain information of flipped images in computation on convolutional layers using discrete cosine transforms. The proposed algorithm on a simple CNN is demonstrated and the efficacy of the proposed algorithm is testified.**

***Index Terms*—convolutional neural network, discrete cosine transform, linear convolution, data augmentation, flipping**

I. Introduction

Convolutional neural networks (CNNs) are widely used in many applications, such as image recognition, image classification, and medical image analysis. AlexNet attracted attention in deep learning [1]. Apart from the number of layers, the architecture of AlexNet is basically the same as that of LeNet [2], which consists of convolutional layers and pooling layers followed by fully connected layers. The environment surrounding deep learning has improved drastically. Now that frameworks for deep learning are in place, it is more difficult to collect a large numbers of training images to improve network performance than to build a CNN.

The performance improves in proportion to the number of learning samples. Data augmentation is a simple but particularly effective method to improve network performance by increasing the number of samples. Generally, samples are generated by applying simple geometric transformation, e.g., rotation, vertical/horizontal translation, scaling, and flipping to training images. There are several data augmentation methods for effectively improving network performance, such as a method using diffuseomorphisms [3], a method using adversarial networks [4], and a method using adaptive generation [5].

In this paper, we focus on flipping in data augmentation, which is effective in simple transformation. We present a novel algorithm that calculates convolution for flipping data augmentation in CNNs. The proposed algorithm obtains not only the convolution of training images but also the convolution of flipped images without generating flipped images beforehand. Use of discrete cosine transforms (DCTs) with zero-padding rather than spatial convolution enables image-free flipping data augmentation. The proposed algorithm is derived by analyzing the relation between linear convolution and circular convolution of DCTs through symmetrically extended sequences. We demonstrate the proposed algorithm on a simple CNN with limited MNIST dataset.

II. Preliminaries

Firstly, we explain leaning of filter coefficients in CNNs briefly. Then, we consider the relation between linear convolution and circular convolution of DCTs through symmetrically extended sequences. One dimensional expression is used for simplicity.

A. Learning of filter coefficients in convolution layers

In CNNs, the filter coefficients (weights of kernel), $\mathbf{h}$, of a convolutional layer are generally updated in iterations of the forward and backward propagation [6] in the learning process as

$$\mathbf{h} \leftarrow \mathbf{h} - \eta \frac{\partial E}{\partial \mathbf{h}} \tag{1}$$

where E denotes the loss function and η is the learning rate.

Let $\mathbf{x}$ and $\mathbf{y}$ be the input feature map and output feature map of a convolutional layer, respectively. In the forward pass, $\mathbf{y}$ is computed as

$$\mathbf{y} = \mathbf{x} * \mathbf{h} \tag{2}$$

where operator '$*$' represents the spatial convolution. In the backward pass, the gradients of E with respect to $\mathbf{x}$ are obtained by

$$\frac{\partial E}{\partial \mathbf{x}} = \frac{\partial E}{\partial \mathbf{y}} * \mathbf{h}^T \tag{3}$$

which is used for the gradients in the previous layer, and the gradients of E with respect to $\mathbf{h}$ are computed by

$$\frac{\partial E}{\partial \mathbf{h}} = \frac{\partial E}{\partial \mathbf{y}} * \mathbf{x}. \tag{4}$$

The spatial convolutions in (2), (3), and (4) can be replaced by the fast discrete Fourier transforms (DFTs), i.e., FFTs for fast training [7]. In this paper, we replace them with DCTs not only for computational complexity but also for flipping data augmentation.

B. The relation between DFT and DCT Type 2 through symmetrically extended sequences

Let $x_N(n)$ be a sequence of length N. The symmetrically extended sequence (SES) of $x_N(n)$ is defined as

$$\hat{x}_{2N}(n) = x_{2N}(n) + x_{2N}(-n-1) \tag{5}$$

where

$$x_{2N}(n) = \begin{cases} x_N(n), & 0 \le n \le N-1 \\ 0, & N \le n \le 2N-1 \end{cases}. \tag{6}$$

The $2N$-point DFT coefficients, $\hat{X}(k)$, of $\hat{x}_{2N}(n)$ are related to the N-point DCT Type 2 (DCT-2) coefficients, $X_C(k)$, of $x_N(n)$ [8], for $k = 0, 1, \ldots, N-1$ as

$$\hat{X}(k) = (1/C_k) X_C(k) W_{2N}^{-k/2} \tag{7}$$

where $W_{2N} = \exp(-j2\pi/2N)$ and

$$\hat{X}(k) = \sum_{n=0}^{2N-1} \hat{x}_{2N}(n) W_{2N}^{nk} \tag{8}$$

$$X_C(k) = 2C_k \sum_{n=0}^{N-1} x_N(n) \cos\left(\frac{\pi(n+1/2)k}{N}\right) \tag{9}$$

$$C_k = \begin{cases} 1/\sqrt{2}, & k = 0 \text{ or } N \\ 1, & \text{otherwise} \end{cases}. \tag{10}$$

Since the DFT has circular periodicity, the relation clearly indicates the symmetrical circular periodicity of DCT-2. This property of DCT-2 is the key to flipping data augmentation.

C. Circular convolution of SESs

The circular convolution of period $2N$ of SESs, $\hat{x}_{2N}(n)$ and $\hat{h}_{2N}(n)$, is calculated using DFT as

$$\hat{y}_{2N}(n) = \frac{1}{2N} \sum_{k=0}^{2N-1} \hat{X}(k)\hat{H}(k) W_{2N}^{-nk} \tag{11}$$

where $\hat{H}(k)$ is the $2N$-point DFT coefficients of $\hat{h}_{2N}(n)$. From (7), $\hat{y}_{2N}(n)$ can be also obtained for $n=0, 1, \ldots, N-1$ by

$$\hat{y}_{2N}(n-1) = \frac{1}{N} \sum_{k=0}^{N} C_k^2 X_C(k) H_C(k) \cos\left(\frac{\pi n k}{N}\right) \tag{12}$$

where $H_C(k)$ is the N-point DCT-2 coefficient of $h_N(n)$ and $Y_C(N) = 0$. Therefore, the first N samples of convolution between SESs is obtained using DCTs without generating SESs.

D. Four linear convolutions in circular convolution of SESs

Let $x(n)$ and $h(n)$ be a sequence of length M and a filter of length L, respectively. We define $x_N(n)$ and $h_N(n)$ of length N as

$$x_N(n) = \begin{cases} x(n-z_1), & z_1 \le n \le z_1 + M - 1 \\ 0, & \text{otherwise} \end{cases} \tag{13}$$

$$h_N(n) = \begin{cases} h(n-z_2), & z_2 \le n \le z_2 + L - 1 \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

where z_1 and z_2 are the number of zeros to be padded before $x(n)$ and $h(n)$, respectively.

The linear convolution, $\hat{y}(n)$, between $\hat{x}_{2N}(n)$ and $\hat{h}_{2N}(n)$ can be expressed by the superposition of four linear convolutions as

$$\hat{y}(n) = y^{(1)}(n) + y^{(2)}(n) + y^{(3)}(n) + y^{(4)}(n) \tag{15}$$

where

$$y^{(1)}(n) = \begin{cases} x(n) * h(n), & l_1 \le n \le l_1 + P - 1 \\ 0, & \text{otherwise} \end{cases} \tag{16}$$

$$y^{(2)}(n) = \begin{cases} x(n) * h(-n-1), & l_2 \le n \le l_2 + P - 1 \\ 0, & \text{otherwise} \end{cases} \tag{17}$$

$$y^{(3)}(n) = \begin{cases} x(-n-1) * h(n), & l_3 \le n \le l_3 + P - 1 \\ 0, & \text{otherwise} \end{cases} \tag{18}$$

$$y^{(4)}(n) = \begin{cases} x(-n-1) * h(-n-1), & l_4 \le n \le l_4 + P - 1 \\ 0, & \text{otherwise} \end{cases} \tag{19}$$

and

$$P = L + M - 1 \tag{20}$$

$$l_1 = z_1 + z_2, \tag{21}$$

$$l_2 = z_1 + 2N - L - z_2, \tag{22}$$

$$l_3 = 2N - M - z_1 + z_2, \tag{23}$$

$$l_4 = 2N - M - z_1 + 2N - L - z_2. \tag{24}$$

$y^{(i)}(n)$, $i = 1, 2, 3, 4$ can be independently obtained by appropriate z_1, z_2 and N [9], which will be used for the convolution for flipping data augmentation using DCTs.

III. The Proposed algorithm

We derive the algorithm for flipping data augmentation using DCTs from the analysis above section.

A. Derivation of convolution for flipping data augmentation using DCTs

Let $x(n_1, n_2)$ and $h(n_1, n_2)$ be an image of size $M \times M$ and a filter of size $L \times L$, repsectively. We define zero-padded images, $x_N(n_1, n_2)$ and $h_N(n_1, n_2)$ as

$$x_N(n_1, n_2) = \begin{cases} x(n_1 - z_1, n_2 - z_1), & R_x \\ 0, & \text{otherwise} \end{cases} \tag{25}$$

$$h_N(n_1, n_2) = \begin{cases} h(n_1 - z_2, n_2 - z_2), & R_h \\ 0, & \text{otherwise} \end{cases} \tag{26}$$

where

$$R_x = \begin{cases} z_1 \le n_1 \le z_1 + M - 1 \\ z_1 \le n_2 \le z_1 + M - 1 \end{cases}, \tag{27}$$

$$R_h = \begin{cases} z_2 \le n_1 \le z_2 + L - 1 \\ z_2 \le n_2 \le z_2 + L - 1 \end{cases}. \tag{28}$$

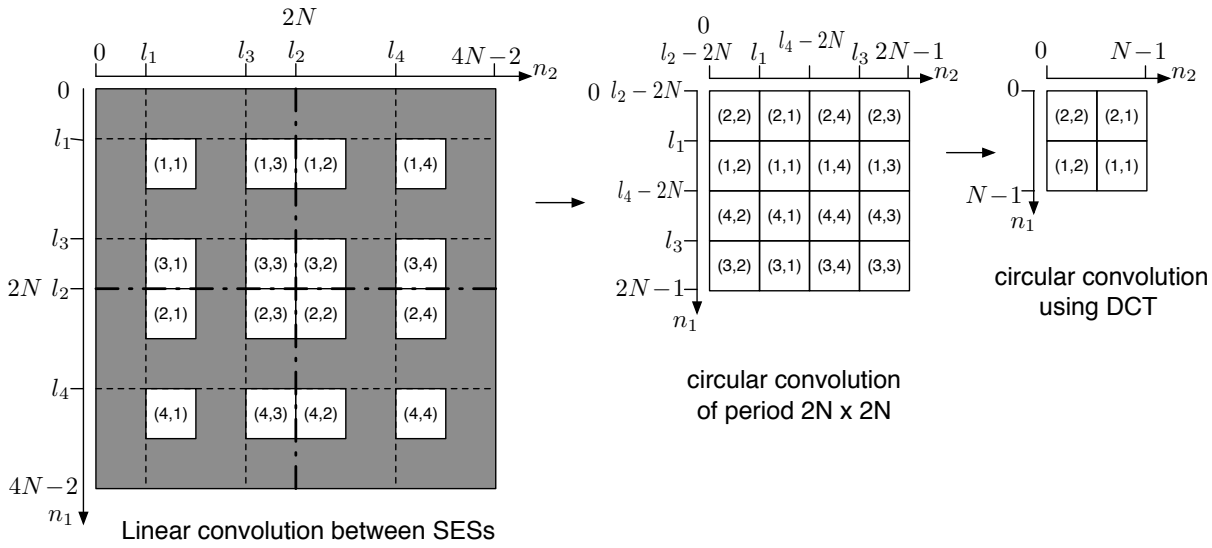

Fig. 1. The concept of convolution for flipping data augmentation using DCTs. The region in white denotes the region of 16 convolutions, $y^{(i,j)}(n_1, n_2)$, where only the superscript (i, j) is indicated.

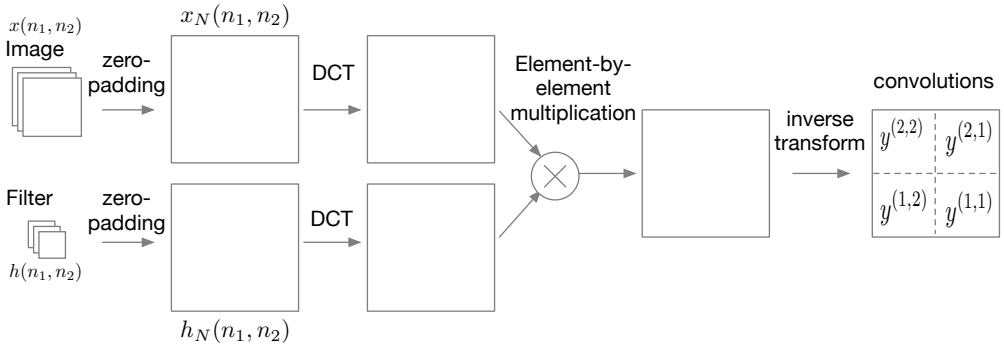

Fig. 2. The proposed algorithm of convolution for flipping data augmentation in a convolutional layer. $y^{(1,1)}$, $y^{(1,2)}$, $y^{(2,1)}$, and $y^{(2,2)}$ represent the linear convolutions of an image, the image flipped in left-and-right, the image flipped in up-and-down, and the image flipped in left-and-right and up-and-down, respectively, with a filter.

The SESs, $\hat{x}_{2N}(n_1, n_2)$ and $\hat{h}_{2N}(n_1, n_2)$, are expressed as

$$\begin{aligned}\hat{x}_{2N}(n_1, n_2) &= x_{2N}(n_1, n_2) + x_{2N}(-n_1, n_2) \\ &\quad + x_{2N}(n_1, -n_2) + x_{2N}(-n_1, -n_2) \qquad (29)\\ \hat{h}_{2N}(n_1, n_2) &= h_{2N}(n_1, n_2) + h_{2N}(-n_1, n_2) \\ &\quad + h_{2N}(n_1, -n_2) + h_{2N}(-n_1, -n_2). \qquad (30)\end{aligned}$$

Therefore, when $\hat{x}_{2N}(n_1, n_2)$ is convolved with $\hat{h}_{2N}(n_1, n_2)$, the output consists of 16 linear convolutions. Each linear convolution is denoted by $y^{(i,j)}(n_1, n_2)$, $i, j = 1, 2, 3, 4$, which corresponds to $y^{(i)}(n)$ with respect to n_1 and $y^{(j)}(n)$ with respect to n_2 in (16) through (19).

In the circular convolution of period $2N \times 2N$, the output samples over $2N \times 2N$ are wrapped around. Under such conditions, from (16) through (19), z_1, z_2 and N are derived so that $y^{(i,j)}(n_1, n_2)$ is isolated as

$$z_1 \geq z_2 + L, \qquad (31)$$

$$z_2 > (M - 2)/2, \qquad (32)$$

$$N \geq z_1 + z_2 + P + 1. \qquad (33)$$

Hence, when using DCTs in (9) and (12), conditions (31), (32), and (33) enable us to obtain four linear convolutions, $y^{(i,j)}(n_1, n_2)$, $i, j = 1, 2$. Fig. 1 illustrates the concept of convolution for flipping data augmentation using DCTs. Zero-padding makes space and isolates latent linear convolutions involved by symmetrical circular periodicity of DCTs.

Note that z_1, z_2, and N decrease when the edges of each linear convolution, $y^{(i,j)}(n_1, n_2)$, are superimposed. The detailed conditions are omitted for space limitation.

B. Computational complexity

DCTs have fast algorithms as the DFT has FFTs. Based on Wang's fast algorithm [10], [11] of DCTs, the computational

TABLE I
COMPUTATIONAL COMPLEXITY OF THE CONVOLUTION FOR FLIPPING DATA AUGMENTATION BETWEEN AN IMAGE OF SIZE $M \times M$ AND A FILTER OF SIZE $L \times L$. WHERE $N = 2(M + L - 1)$ FOR FULL VERSION.

forward transform	mu.	$2 \times (N^2 \log_2 N + 2N)$
	ad.	$2 \times (3N^2 \log_2 N - 2N^2 + 2N)$
product	mu.	N^2
inverse transform	mu.	$1 \times (N^2 \log_2 N + 2N)$
	ad.	$1 \times (3N^2 \log_2 N - 2N^2 + 2N)$

TABLE II
TEST ACCURACY [%] FOR CLASSIFYING HANDWRITTEN DIGITS IN LIMITED MNIST.

algorithm	test acc.
spatial conv.	95.3
spatial conv. with DA (flipping)	92.8
spatial conv. with zero-padding	95.3
proposed (limited)	96.7
proposed (full)	97.3

complexity of convolution between an image and a filter for flipping data augmentation is summarized in Table I.

Let S and F be the size of minibatches and number of filters, respectively. Spatial convolution needs $SF(M - L + 1)^2L^2$ multiplications, while the proposed algorithm requires $S(2(N^2 \log_2 N + 2N) + N^2) + F(N^2 \log_2 N + 2N)$ multiplications, where $N = 2(M + L - 1)$. For example, when $S = 100$, $F = 30$, $M = 28$, and $L = 5$, spatial convolution needs 43,200,000 multiplications, while the proposed algorithm requires 6,091,520 multiplications. Thus, in a convolutional layer, the operations of the proposed algorithm is less than spatial convolution, although the size of output feature map is larger.

Practically, in CNNs, once the DCT-2 coefficients of filters are obtained, the coefficients are reused unless the coefficients are updated, which reduces the number of operations. Moreover, in learning, zero-padding and applying DCT-2 to training images as preprocessing also reduce the number of operations.

C. Steps of the proposed algorithm

In the forward pass, we pad zero-values to images and filters in a convolutional layer according to (25), (26), (31), (32), and (33). The DCT-2 is applied to the zero-padded images and filters according to (9). Then, DCT-2 coefficients are multiplied element-by-element. Finally, the inverse transform is applied to the product to obtain the convolution for flipping data augmentation according to (12). In the backward pass, there is no need for zero-padding. Fig. 2 shows the proposed algorithm of convolution for flipping data augmentation in a convolutional layer.

IV. EXPERIMENTAL RESULTS

We show the performance of proposed algorithm using limited MNIST dataset.

A. Experimental setup

We compared the proposed algorithm to spatial convolution, spatial convolution with flipping data augmentation, and spatial convolution with zero-padding. In spatial convolution with flipping data augmentation, 15,000 images were newly generated by flipping 5,000 training images in left-and-right, up-and-down, and both of them, i.e., a total of 20,000 images were used for training. In spatial convolution with zero-padding, the zero-values were padded to training images so that the output feature map becomes the same size as that of the proposed algorithm. In the proposed algorithm, whole convolution was used as the output feature map (full version) and part of convolution was used as the output feature map (limited version).

The network for evaluation consists of a convolutional layer, a pooling layer, two fully connected layers followed by the softmax layer, where rectified linear unit (ReLU) activation functions were used after the convolutional layer and the fully-connected layer. The convolutional layer have 30 filters of size 5×5. Adam [12] was used as the optimizer. The cross entropy error was employed as the loss function. The initial filter coefficients were randomly set. The configuration had minibatches of size 100. Each configuration was trained for 20 epochs with η=0.001. We used MNIST dataset that consists of 60,000 handwritten digits images of size 28×28 and 10,000 test images. We limited the training images to 5,000 for evaluation.

B. Experimental results

Table II summarizes the test accuracy of the different settings. Since the digits are not flipped in test images, the accuracy of spatial convolution with flipping data augmentation was worst (acc. = 92.8 %). The accuracy of spatial convolution and spatial convolution with zero-padding was the same, 95.3 %. The proposed algorithm (full version) achieved the best accuracy, 97.3%. Considering that the digits are not flipped left and right in test images, the increase in accuracy by the proposed algorithm can be expected more in general images including flipped patterns, especially left and right flipping in natural images. We also confirmed that the accuracy of the CNN with the proposed algorithm increases fast in learning process.

V. CONCLUSIONS

We proposed a novel algorithm of flipping data augmentation in a CNN. Use of DCTs with zero-padding rather than spatial convolution enables image-free flipping data augmentation, i.e., without preparing flipped images beforehand, the convolutions of such images with filters are calculated in CNNs. We demonstrated the efficacy of the proposed algorithm on a simple CNN to classify handwritten digits using limited MNIST dataset. In the future work, we will evaluate the algorithm in deep CNNs using general images and measure the learning time with GPU implementations.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, 2012, pp. 1097–1105.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, November 1998, pp. 2278–2324.

[3] S. Hauberg, O. Freifeld, A. B. L. Larsen, J. W. F. III, and L. K. Hansen, "Dreaming more data: Class-dependent distributions over diffeomorphisms for learned data augmentation," in *Proceedings of the 19th international Conference on Artificial Intelligence and Statistics (AISTATS)*, 2016.

[4] X. Zhang, Z. Wang, D. Liu, and Q. Ling, "Dada: Deep adversarial data augmentation for extremely low data regime classification," in *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019, pp. 2807–2811.

[5] A. Fawzi, H. Samulowitz, D. Turaga, and P. Frossard, "Adaptive data augmentation for image classification," in *International Conference on Image Processing*. IEEE, 2016, pp. 3688–3692.

[6] Y. LeCun, L. Bottou, G. Orr, and K. Muller, *Neural Networks: Tricks of the trade*. Springer, 1998.

[7] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," in *International Conference on Learning Representations*. arXiv:1312.5851, 2014.

[8] A. V. Oppenheim, R. W. Sshafer, and J. R. Buck, *Discrete-time signal processing*. New Jersey: Prentice Hall, Inc., 1999.

[9] I. Ito, "A computing method for linear convolution and linear correlation in the dct domain," *IEICE Trans. Fundamentals*, vol. E96-A, no. 7, pp. 1518–1525, July 2013.

[10] Z. Wang, "Fast algorithms for the discrete w transform and for the discrete fourier transform," vol. 32, no. 4, pp. 803–816, 1984.

[11] ——, "On computing the discrete fourier and cosine transforms," vol. 33, no. 4, pp. 1341–1344, 1985.

[12] D. Kingma and J. Ba., "A method for stochastic optimization," in *arXiv:1412.6980*, 2014.