CSIAM Trans. Appl. Math.
doi: 10.4208/csiam-am.2020-0018

Vol. **1**, No. 3, pp. 518-529
September 2020

Hartley Spectral Pooling for Deep Learning

Hao Zhang[1] and Jianwei Ma[1,2,*]

[1] *Department of Mathematics and Artificial Intelligence Laboratory, Harbin Institute of Technology, Harbin 150001, China.*
[2] *School of Earth and Space Sciences, Peking University, Beijing 100000, China.*

Received 5 May 2020; Accepted 12 September 2020

Abstract. In most convolution neural networks (CNNs), downsampling hidden layers is adopted for increasing computation efficiency and the receptive field size. Such operation is commonly called pooling. Maximization and averaging over sliding windows (*max/average pooling*), and plain downsampling in the form of strided convolution are popular pooling methods. Since the pooling is a lossy procedure, a motivation of our work is to design a new pooling approach for less lossy in the dimensionality reduction. Inspired by the spectral pooling proposed by Rippel et al. [1], we present the Hartley transform based spectral pooling method. The proposed spectral pooling avoids the use of complex arithmetic for frequency representation, in comparison with Fourier pooling. The new approach preserves more structure features for network's discriminability than max and average pooling. We empirically show the Hartley pooling gives rise to the convergence of training CNNs on MNIST and CIFAR-10 datasets.

AMS subject classifications: 68T07

Key words: Hartley transform, spectral pooling, deep learning.

1 Introduction

Convolutional neural networks(CNNs) [2–4] have been dominant machine learning approach for computer vision, and have spreaded out in many other fields. The modern framework of CNNs was established by LeCun et al. [5] in 1990, with three main components: convolution, pooling, and activation. Pooling is an important component of CNNs. Even before the resuscitation of CNNs, pooling was utilized to extract features to gain dimension-reduced feature vectors and acquire the invariance to small transformations of the input. This is motivated by the seminal work about complex cells in animal visual cortex by Hubel and Wiesel [6].

*Corresponding author. *Email addresses:* jwm@pku.edu.cn (J. Ma), hao.zhang.hit@stu.hit.edu.cn (H. Zhang)

Pooling is of crucial for reducing computation cost, improving some amount of translation invariance and increasing the receptive field of neural networks. Numerous variants of pooling processes are proposed for classification accuracy improvement. These variants are mainly casted in four major categories based on value, rank, probability and transformed domain pooling methods, which are thoroughly reviewed recently in [32]. In shallow or mid-sized networks, max or average pooling are most widely used such as in AlexNet [7], VGG [8], and GoogleNet [9]. After Springenberg et al. [10] empirically revealed that strided convolution could replace pooling without loss of accuracy in classification task, deeper networks always use strided convolution for architecture-design simplicity. The most markable one of those exemplars is ResNet [11]. However, most methods present a number of issues. For example, max pooling implies an amazing by-product of discarding at least 75% of data. It can overfit the training data and does not guarantee generalization on test data. The maximum value picked out in each window only reflects very rough information. Average pooling, stretching to the opposite end, results in a gradual, constant attenuation of the contribution of individual grid in each window, and ignores the importance of local structure. These two poolings both suffer from sharp dimensionality reduction and lead to implausible looking results (see the first and second row in Fig. 1). Strided convolution may cause aliasing since it simply picks one node in a fixed position in each local window [12], regardless of the significance of its activation.

There have been a few attempts to mitigate the harmful effects of max and average pooling, such as a linear combination and extension of them [13], and nonlinear pooling layers [14, 15]. In most of the common implementations, max or average related pooling layers directly downscale the spatial dimension of feature maps by a factor. L_p pooling [15] provides better generalization than max pooling, with $p=1$ corresponding to average pooling and $p=\infty$ reducing to max pooling. Yu et al. [16] proposed the mixed pooling, which combines max pooling and average pooling and switches between these two pooling methods randomly. Instead of picking the maximum values within each pooling region, stochastic pooling [17] and S3Pool [18] stochastically pick a node in a window, and the former favors strong activations. In some networks, stride convolutions are also used for pooling. Notably, these pooling methods are all of integer stride larger than 1. To abate the loss of information caused by the dramatic dimension reduction, fractional max-pooling [19] randomly generates pooling region with stride 1 or 2 to achieve pooling stride of less than 2. There are also some other pooling methods by applying/learning filters. For example, detail preserving pooling [33] uses inverse bilateral filter and learns two reward parameters in inverse bilateral weights from data to adaptively preserve the important details of feature maps. LEAP pooling [34] learns a shared linear filter over feature maps and aggregates the features within pooling region. We refer these pooling methods mentioned above to as *spatial pooling*.

In 2015, Rippel et al. [1] proposed the *spectral pooling*, which downsamples the feature maps in frequency domain using low-pass filtering. It selects pooling region in Fourier based frequency domain by extracting low frequency subset. This approach can allevi-

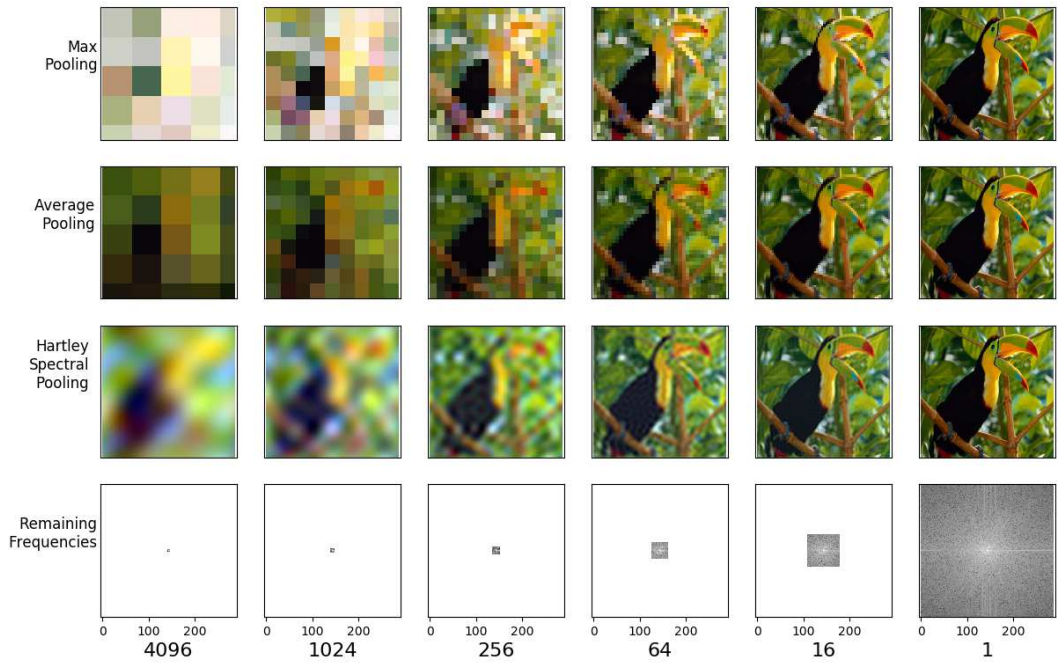

Figure 1: Downsampling at different scales of dimensionality reduction. Hartley-based spectral pooling project real input onto the Hartley basis and truncates the real frequency representation as desired. This retains significantly more information as well as allows us choose arbitrary output dimension.

ates the issues of spatial pooling strategies as mentioned above, and it shows good information preserving ability. However, it introduces the processing of imaginary, which should be carefully treated in real CNNs. Moreover, the truncation in frequency domain may destroy the *conjugate symmetry* of the Fourier frequency representation of the real input. For remediation some extra operations should be added to make sure the downsampled spatial approximation be real. This would be tedious and sometimes computation/time consuming. Following the work of [1], spectral pooling approach using discrete cosine transform (DCT) [35] and wavelet transform [36] are also proposed.

Inspired by the work of [1], we present the Hartley transform-based spectral pooling in this paper. Our presented approach avoids the use of complex arithmetic and it could be plugged in modern CNNs effortlessly. Moreover, we provide a useful observation that preserving more information could contribute to the convergence of training modern CNNs.

2 Method

2.1 Hartley transform

The Hartley transform is an integral transform closely related to the Fourier transform [20, 21]. It has some advantages over the Fourier transform in the analysis of real sig-

nals as it avoids the use of complex arithmetic. These advantages attracts researchers to conduct plenty of researches on its application and fast implementation during the 1990s [22–25].

In two dimensions, the Hartley transform, $H(u_1,u_2)$, of $f(x_1,x_2)$ is defined by [24]

$$H(u_1,u_2)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x_1,x_2)[\cos(2\pi(u_1x_1+u_2x_2)) +\sin(2\pi(u_1x_1+u_2x_2))]dx_1dx_2, \tag{2.1}$$

and the inverse transform by

$$f(x_1,x_2)=\int_{-\infty}^{\infty}H(u_1,u_2)[\cos(2\pi(u_1x_1+u_2x_2)) +\sin(2\pi(u_1x_1+u_2x_2))]du_1du_2. \tag{2.2}$$

We write the kernel in the Hartley transform as

$$\begin{aligned} cas(x)&=\cos(x)+\sin(x)\\ &=\frac{1-i}{2}e^{ix}+\frac{1+i}{2}e^{-ix}, \end{aligned} \tag{2.3}$$

so that

$$e^{ix}=\frac{1+i}{2}cas(x)+\frac{1-i}{2}cas(-x). \tag{2.4}$$

Then the relation between the Hartley and Fourier transform can be derived from the above expressions, giving

$$H(u_1,u_2)=\frac{1+i}{2}F(u_1,u_2)+\frac{1-i}{2}F(-u_1,-u_2) \tag{2.5}$$

and

$$F(u_1,u_2)=\frac{1-i}{2}H(u_1,u_2)+\frac{1+i}{2}H(-u_1,-u_2), \tag{2.6}$$

where $F(u_1,u_2)$ represents the Fourier transform of $f(x_1,x_2)$. In the case that function $f(x_1,x_2)$ is real, its Fourier transform is Hermetian, i.e.

$$F(-u_1,-u_2)=F^*(u_1,u_2), \tag{2.7}$$

so the Fourier transform processes some redundancy on the real u_1-u_2 plane, which results in conjugate-symmetry constriction aiming at reducing training parameters in the frequency domain neural networks [1,26].

The Hermetian property above shows that the Hartley transform of a real function can be written as

$$H(u_1,u_2)=\mathcal{R}\{F(u_1,u_2)\}-\mathcal{I}\{F(u_1,u_2)\}, \tag{2.8}$$

where $\mathcal{R}\{\cdot\}$ and $\mathcal{I}\{\cdot\}$ denote the real and imaginary parts respectively. Note that given the definition above, the Hartley transform $\mathcal{H}$ is a real linear operator. It is symmetric, an involution and thus a unitary operator

$$f=\mathcal{H}\{\mathcal{H}f\}. \tag{2.9}$$

In the case of Hartley transform, imaginary part and conjugate symmetry no more need concerns for real inputs such as images. Note, moreover, that this does not increase any storage.

Differentiation. Here we discuss how to propagate the gradient through the Hartley transform, which will be used in CNNs. Define $x\in\mathbb{R}^{M\times N}$ and $y=\mathcal{H}(x)$ to be the input and output of a discrete Hartley transform (DHT) respectively, and $L:\mathbb{R}^{M\times N}\to\mathbb{R}$ a real-valued loss function applied to y. Since the DHT is a linear operator, its gradient is simply the transform matrix itself. By the unitarity of DHT, the gradient in back-propagation corresponds to applying Hartley transform :

$$\frac{\partial L}{\partial x}=\mathcal{H}\Big(\frac{\partial L}{\partial y}\Big). \tag{2.10}$$

2.2 Hartley-based spectral pooling

Spectral pooling preserves considerably more information and structures for the same number of parameters [1], as shown in the third row of Fig. 1. This is because spectral transform provides a sparse basis in frequency domain for the inputs that have spatial structures. The spectrum power of a typical input is heavily concentrated in lower frequencies while higher frequencies mainly tend to encode noise [27]. This non-uniformity of spectrum power enables the removal of high frequencies with minimal damage of input information.

To avoid the extra computation for conjugate symmetry ensurance which may be computation and time consuming in [1], we suggest the Hartley transform-based spectral pooling. This spectral pooling is straightforward to understand and much easier to implement. Assume we have an input $x\in\mathbb{R}^{H\times W}$, and some desired output map dimensionality $h\times w$. First, we compute the DHT of the input into the frequency domain as $y=\mathcal{H}(x)\in\mathbb{R}^{H\times W}$, and shift the DC component of the input to the center of the domain. Then we crop the frequency representation by maintaining only the central $h\times w$ submatrix of frequencies, denoted as $\hat{y}\in\mathbb{R}^{h\times w}$. Finally, we take the DHT again as $\hat{x}=\mathcal{H}(\hat{y})$ to map frequency approximation back into spatial domain, obtaining the downsampled spatial approximation. The back-propagation of this spectral pooling is similar to its forward-propagation since Hartley transform is differentiable.

Those steps in both forward and backward propagation of this spectral pooling are listed in Algorithm 1 and 2, respectively. These algorithms simplify the spectral pooling by Fourier transform, profited from that the Hartley transform of a real function is real

rather than complex. Fig. 1 demonstrates the effect of this spectral pooling for various dimensionality reduction factors.

Algorithm 1 Hartley Spectral pooling

Require: Map $x \in \mathbb{R}^{H\times W}$, output size $h \times w$
Ensure: Pooled map $\hat{x} \in \mathbb{R}^{h\times w}$
1: $y \leftarrow \mathcal{H}(x)$
2: $\hat{y} \leftarrow CropSpectrum(y, h \times w)$
3: $\hat{x} \leftarrow \mathcal{H}(\hat{y})$

Algorithm 2 Hartley Spectral pooling back-propagation

Require: Gradients w.r.t. output $\frac{\partial L}{\partial \hat{x}}$
Ensure: Gradients w.r.t. input $\frac{\partial L}{\partial x}$
1: $\hat{z} \leftarrow \mathcal{H}(\frac{\partial L}{\partial \hat{x}})$
2: $z \leftarrow PadSpectrum(\hat{z}, H \times W)$
3: $\frac{\partial L}{\partial x} \leftarrow \mathcal{H}(z)$

3 Experiments

We verify the effectiveness of the Hartley-based spectral pooling through image classification task on MNIST, Fashion-MNIST and CIFAR-10 datasets. The trained networks include a toy CNN model (Table 1), ResNet-16 and ResNet-20 [11]. The toy network uses max pooling while ResNets employs strided convolutions for downscaling. In these experiments, spectral pooling shows favorable results. We also compare our DHT spectral pooling method with the DCT spectral pooling method [35]. Our implementation is based on PyTorch [28].

3.1 Datasets and configurations

MNIST. The MNIST database [29] is a large database of handwritten digits that is commonly used for benchmarking various convolutional neural networks. This dataset contains 60000 training examples and 10000 testing examples. All these examples are gray images in size of 28×28. In our experiment, we do not perform any preprocessing or augmentation on this dataset. Adam optimization algorithm [30] is used in all experiments of classification on MNIST, with hyper-parameter $\beta_1 = 0.9$ and $\beta_2 = 0.999$ configured as suggested and a mini-batch size of 100. The initial learning rate is set to 0.001 and is divided by 10 every 5 epochs. Regularization is aborted in these experiments.

Figure 2: 500 examples in MNIST dataset. From top row to bottom labels 0 to 9 respectively.

Fashion-MNIST. The Fashion-MNIST [37] dataset shares the same image size, data format and the structure of training and testing splits with MNIST. But it is a more challenging alternative dataset for benchmarking machine learning methods since MNIST dataset is too easy. If the methods do work on MNIST, they may still fail on others. In our experiments, we set the training configurations same as that in MNIST.

CIFAR-10. The CIFAR-10 dataset consists of 60000 colored natural images in 10 classes, with 6000 images per class holding 5000 for training and 1000 for testing. Each image is in size of 32×32. For data augmentation we follow the practice in [11], doing horizontal flips and randomly sampling 32×32 crops from image padded by 4 pixels on each side. The normalization is performed in data preprocessing by using the channel means and standard deviations. In experiments on this dataset, we use stochastic gradient method with Nesterov momentum and cross-entropy loss. The initial learning rate is set to 0.1, and is multiplied by 0.1 at 80 and 120 epochs. The weight decay parameter is configured to 10^{-4} and the momentum is set to 0.9 without dampening. Mini-batch size is set to 128.

3.2 Classification results on MNIST and Fashion-MNIST

Shallow network. We first conduct experiments using the toy network (see Table 1) on MNIST. Each convolution layer is followed by a batch normalization and a ReLU nonlinearity. We test Hartley-based spectral pooling by replacing max pooling in this architecture. The training procedure lasts 10 epoches and is repeated 10 times. The classification error on testing set in each epoch is shown in Fig. 3.

Compared to max pooling, spectral pooling shows strong results, yielding more than 15% reduction on classification error observed in this experiment. As all things equal except the pooling layers, we claim that this improvement is achieved by the better information-preserved ability of Hartley-based spectral pooling.

ResNet. Next, we conduct experiments on both MNIST and Fashion-MNIST datasets using ResNet-20 [11]. We don't use more deeper residual net such as ResNet-110 because much more parameters in this architecture may give rise to overfitting. ResNet is composed of numerous residual building blocks. It is a much modern convolutional neural network which does not explicitly use pooling layers but instead embeds a stride-2 convolution layer inside some of building blocks for the implementation of downsampling.

Table 1: The toy CNN model for classification on MNIST.

layer name	output size	Max Pooling model	Spectral Pooling model
conv1	28×28	5×5, 16	
pool1	14×14	Max, stride=2	Spectral, 14×14
conv2	14×14	5×5, 32	
pool2	7×7	Max, stride=2	Spectral, 7×7
fc	1×1	10-d fc	

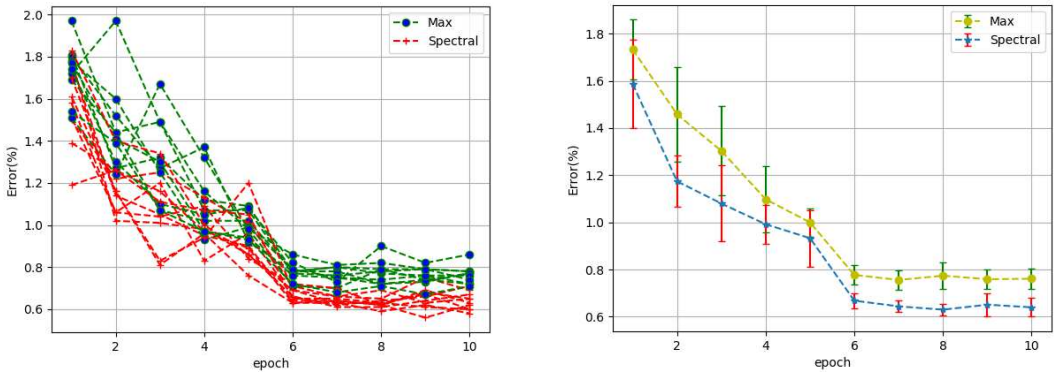

Figure 3: Classification error on MNIST testing set by networks in Table 1. (**Left**) Classification error curves in 10 runs. (**Right**) *mean*±*std* of ten runs, with best error 0.605% (0.63 ± 0.025), 0.719% (0.759 ± 0.040) for spectral pooling and max pooling respectively.

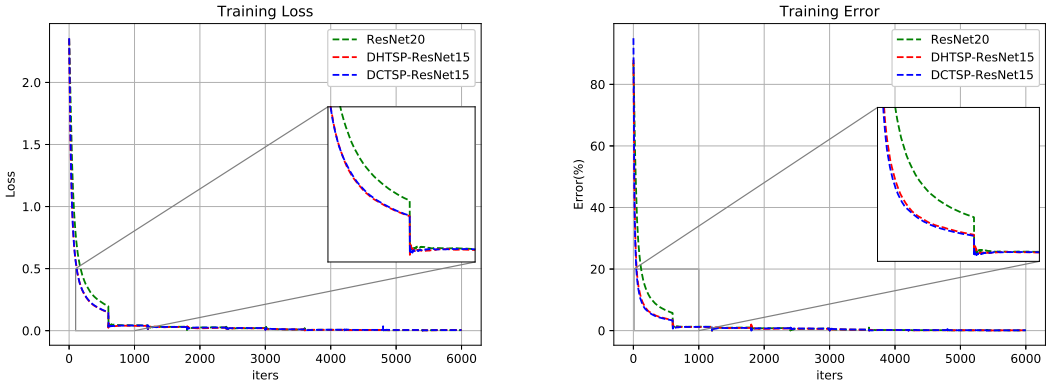

Figure 4: Training on **MNIST** by ResNet20 and SP-ResNet15. (**Left**) training loss; (**Right**) classification error on training set.

In our experiments we replace the stride-2 convolutional layer by spectral pooling and remove the skip connection [9, 31] in those downscaling blocks. Besides, we set the output size of serial spectral pooling layers linearly decreased (reducing 8 in each axis after a spectral pooling layer). The manually tuned network architecture is depicted in Table 2.

Table 2: The architecture of spectral pooling ResNet for MNIST and Fashion-MNIST. Building blocks are shown in brackets, with the numbers of block stacked. The SP stands for the spectral pooling layer, and the footnote $n\times n$ indicates the output size.

block name	output size	SP-ResNet15
conv1	28×28	3×3, 16
conv2	28×28	$\begin{bmatrix}3\times3,16\\3\times3,16\end{bmatrix}\times2$
downsample1	20×20	$\begin{bmatrix}SP_{20\times20}\\3\times3,32\end{bmatrix}$
conv3	20×20	$\begin{bmatrix}3\times3,32\\3\times3,32\end{bmatrix}$
downsample2	12×12	$\begin{bmatrix}SP_{12\times12}\\3\times3,32\end{bmatrix}$
conv4	8×8	$\begin{bmatrix}3\times3,32\\3\times3,32\end{bmatrix}$
downsample3	4×4	$\begin{bmatrix}SP_{4\times4}\\3\times3,64\end{bmatrix}$
conv5	4×4	$\begin{bmatrix}3\times3,64\\3\times3,64\end{bmatrix}$
	1×1	avg. pool, 10-d fc

We leave the global average pooling untouched. Each run is repeated 5 times. The number of parameters in each network, the averaged training time of single epoch, the best result as well as mean and standard deviation are reported in Table 3. To distinguish the different spectral pooling methods, we prefix SP-ResNet15 with the spectral transform name (DHT, DCT).

As shown in Table 3, the DHTSP-ResNet15 obtains slightly better testing error (+0.04%) than the plain ResNet20 in MNIST task, and a significant improvement (+0.65%) in Fashion-MNIST task, even though it holds parameters nearly half of that in ResNet20. The training time increases 4.2 s for a single epoch with the replacement of strided convolution by DHT spectral pooling. The DCTSP-ResNet15 performs slightly better (+0.14%) than our DHTSP-ResNet15 on Fashion-MNIST dataset, but the training time doubles. The slightly outperforming of the DCTSP over our DHTSP is not surprising because for DCT more energy is concentrated in even fewer spectra. Further, we illustrate one among the five training procedures in Fig. 4. It is observed that the SP-ResNet15 converges faster than ResNet20 (left panel) and performs better in classification (right panel). This indicates the spectral pooling eases the optimization by providing faster convergence at the early stage.

Table 3: Classification error on **MNIST** (left) and **Fashion-MNIST** (right) testing set. All methods are without data augmentation. The average training time of a single epoch is reported. For each method we run it 5 times and show "best(mean±std)".

method	#params	training time (s)	error (%)
ResNet16	0.18M	25.2	0.36 (0.40±0.04)/ 7.14 (7.23±0.06)
ResNet20	0.27M	26.4	0.36 (0.40±0.04)/6.91(7.12±0.13)
DHTSP-ResNet15	**0.15M**	30.6	**0.32** (0.37±0.04)/6.26 (6.38± 0.07)
DCTSP-ResNet15		66.7	**0.32** (0.38±0.04)/**6.12** (6.21±0.07)

3.3 Classification results on CIFAR-10

For the experiments on CIFAR-10, the architecture of spectral pooling ResNet is similar to SP-ResNet15 that is used in MNIST case. We set the sizes of output of spectral pooling layers to be 24×24, 16×16, 8×8 sequentially. We use plain ResNet-16 as a counterpart, since it contains almost the same amount of parameters as SP-ResNet15 (see Table 4). The DHTSP-ResNet15 outperforms ResNet16 with smaller best testing error by 0.24% and smaller mean testing error by 0.1%, although the training time increases about 1000 s. The DCTSP-ResNet15 performs slightly worse than DHTSP-ResNet15, and doubles the training time. It gets even slightly worse mean testing error than the plain ResNet-16.

Table 4: Classification error on **CIFAR-10** testing set. All methods are with data augmentation. For each method we run it 5 times and show "best(mean±std)".

method	# parameters	training time (s)	error (%)
ResNet16	0.18M	2896.4	8.87 (9.06±0.10)
DHTSP-ResNet15	**0.15M**	3900.8	**8.63** (8.96±0.16)
DCTSP-ResNet15		7808.5	8.81 (9.11±0.17)

4 Conclusion

We present a full real-valued Hartley spectral pooling method in this paper. Based on this approach, we provide some results on several commonly used benchmark dataset (MNIST, Fashion-MNIST and CIFAR-10) by training a toy CNN and the modified Resnets. We demonstrate the Hartley spectral pooling yields higher classification accuracy than its counterpart max pooling. We also investigate the contribution of this spectral pooling method to the convergence of training neural networks. In Resnet, it improves the results by expanding the space of spatial dimensionality of downsamplings. We also compare the Hartley spectral pooling with its spectral pooling counterpart, the discrete cosine transform spectral pooling. Although sometimes the Hartley spectral pooling perform slightly worse than the discrete cosine transform spectral pooling in respect of classification accuracy, it is much faster. Moreover, the Hartley transform obeys

an analogous convolution theorem, which means it has the potential for constructing full real spectral neural nets.

Acknowledgments

The work is supported in part by National Key Research and Development Program of China under Grant 2017YFB0202902, and NSFC under Grant 41625017.

References

[1] O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 2449-2457, 2015.
[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. IEEE*, 86(11): 2278-2324, 1998.
[3] J. Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61: 85-117, 2015.
[4] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553): 436-444, 2015.
[5] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel. Handwritten digit recognition with a backpropagation network. In *Advances in Neural Information Processing Systems*, pp. 396-404, 1990.
[6] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of Physiology*, 160(1): 106-154, 1962.
[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pp. 1097-1105, 2012.
[8] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.
[9] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich et al.. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.
[10] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. Striving for simplicity: The all convolutional net. In *International Conference on Learning Representations*, 2015.
[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770-778, 2016.
[12] J. G. Proakis and D. G. Manolakis. Digital Signal Processing. *Pearson Education*, 2013.
[13] C.-Y. Lee, P. W. Gallagher, and Z. Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. *Artificial Intelligence and Statistics*, pp. 464-472, 2016.
[14] C. Gulcehre, K. Cho, R. Pascanu, and Y. Bengio. Learned-norm pooling for deep feedforward and recurrent neural networks. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 530-546, 2014.
[15] J. Bruna, A. Szlam, and Y. LeCun. Signal recovery from pooling representations. In *International Conference on Machine Learning*, pp. 1585-1598, 2014.
[16] D. Yu, H. Wang, P. Chen, and Z. Wei. Mixed pooling for convolutional neural networks. In *International Conference on Rough Sets and Knowledge Technology*. Springer, pp. 364-375, 2014.
[17] M. D. Zeiler and R. Fergus. Stochastic pooling for regularization of deep convolutional neural networks. In *International Conference on Learning Representations*, 2013.

[18] S. Zhai, H. Wu, A. Kumar, Y. Cheng, Y. Lu, Z. Zhang, and R. Feris. S3pool: Pooling with stochastic spatial sampling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4970-4978, 2017.
[19] B. Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014.
[20] R. V. Hartley. A more symmetrical fourier analysis applied to transmission problems. *Proceedings of the IRE*, 30(2): 144-150, 1942.
[21] R. N. Bracewell. The Hartley Transform. *Oxford University Press, Inc.*, 1986.
[22] R. N. Bracewell. Discrete Hartley transform. *J. Opt. Soc. Am.*, 73: 1832-1835, 1983.
[23] R. N. Bracewell. The fast hartley transform. *Proceedings of the IEEE*, 72(8): 1010-1018, 1984.
[24] R. Millane. Analytic properties of the hartley transform and their implications. *Proceedings of the IEEE*, 82(3): 413-428, 1994.
[25] J. Agbinya. Fast interpolation algorithm using fast hartley transform. *Proceedings of the IEEE*, 75(4): 523-524, 1987.
[26] H. Pratt, B. Williams, F. Coenen, and Y. Zheng. FCNN: Fourier convolutional neural networks. *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, pp. 786-798, 2017.
[27] A. Torralba and A. Oliva. Statistics of natural image categories. *Network: Computation in Neural Systems*, 14(3): 391-412, 2003.
[28] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *Advances in Neural Information Processing Systems*, 2017.
[29] E. Kussul and T. Baidyk. Improved method of handwritten digit recognition tested on mnist database. *Image and Vision Computing*, 22(12): 971-981, 2004.
[30] D. P. Kingma and J. Ba. Adam. A method for stochastic optimization. In *International Conference on Learning Representations*, 2015.
[31] R. K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. In *International Conference on Machine Learning Deep learning workshops*, 2015.
[32] N. Akhtar and U. Ragavendran. Interpretation of intelligence in CNN-pooling processes: A methodological survey. *Neural Comput. & Applic.*, 32: 879898, 2020.
[33] F. Saeedan, N. Weber, M. Goesele, and S. Roth. Detail-preserving pooling in deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9108-9116, 2018.
[34] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang. Learning pooling for convolutional neural network. *Neurocomputing*, 224: 96-104, 2017.
[35] J. S. Smith and B. M. Wilamowski. Discrete cosine transform spectral pooling layers for convolutional neural networks. In *International Conference on Artificial Intelligence and Soft Computing*, pp. 235-246. Springer, Cham, 2018.
[36] T. Williams and R. Li. Wavelet pooling for convolutional neural networks. In *International Conference on Learning Representations*, 2018.
[37] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.