JOURNAL OF INFORMATION SCIENCE AND ENGINEERING 36, 1069-1078 (2020)
DOI: 10.6688/JISE.202009_36(5).0009

Convolutional and Fully Connected Layer in DFN

MIAN MIAN LAU AND KING HANN LIM
Department of Electrical and Computer Engineering
Curtin University Malaysia
CDT 250, 98009 Miri, Sarawak, Malaysia
E-mail: mian.lau@postgrad.curtin.edu.my

Deep feedforward network (DFN) is the general structure of many well-known deep neural networks (DNN) for image classification. The recent research emphasizes on going deeper and wider network architecture to achieve higher accuracy and lower misclassification rate. This paper provides a study and investigation on stacking three basic operation of neural layers, *i.e.* convolutional layer, pooling layer and fully connected layer. As a result, a new framework of convolutional deep feedforward network (C-DFN) is proposed in this paper. C-DFN performed significantly better than deep feedforward network (DFN), deep belief network (DBN), and convolutional deep belief network (C-DBN) in MNIST dataset, INRIA pedestrian dataset and Daimler pedestrian dataset. The convolutional layer acts as a trainable feature extractor improving the network performance significantly. Moreover, it reduced 14% of the trainable parameters in DFN. With the use of trainable activation function such as PReLU in the C-DFN, it achieves an average misclassification rate of 9.22% of the three benchmark datasets.

Keywords: convolution network, deep feedforward network, fully connected network, stacking effect, convolutional deep belief network

1. INTRODUCTION

Deep Feedforward Network (DFN) is the fundamental network architecture in the recent development of deep learning neural networks for image classification. DFN is a computer algorithm that mimics biological neural network to solve complicated real-world application. Going deeper and wider network architecture has attracted large research attention due to the improvement of classification accuracy. A multiple layers deep structures neural networks (DNN) can be applied in many complicated task such as object recognition [1], speech recognition [2], and handwritting recognition [3]. The development of DNN has been exponentially growing after AlexNet [4] showed a drastic drop of top-5 error from 26.2% to 15.4% in the ImageNet large scale visual recognition challenges [5]. In the subsequent years, DNNs such as ZF net [6], VGG net [7], and GoogLeNet [8] was developed in deeper and wider neural networks to compete in ImageNet Challenge. Microsoft research team [9] developed the largest DNN structure containing 152 layers and hence it achieved 3.6% of misclassification rate in the challenge. However, the study

Received August 11, 2019; revised October 25, 2019; accepted November 21, 2019.
Communicated by Wei Kitt Wong.

of very deep network architecture always associates with the stacking of network layers and involvement of large computing resources.

Klambauer *et al*. [10] introduced self-normalizing to normalize the output of activation function with the convergence of zero mean and unit variance. This convergence property allows fully-connected neural network (FCN) to be trained in multiple layers without having vanishing and exploding gradient problems. Hinton *et al*. [11] proposed a layer-wise pre-training algorithm in deep belief network to solve the vanishing gradient problem. Other FCN architectures such as dropout network [12], maxout network [13], and dropConnect network [14] solved the overfitting problems of the FCN. Lin *et al*. [15] proposed network in network (NiN) combining convolutional and fully connected network as a micro network in NiN architecture. In the micro-network, it consists of convolutional layer (CL), pooling layer (PL) and the fully connected network (FL) which act as a potent function approximator. As a result, NiN is able to reduce the test error of CIFAR-10 and CIFAR-100 to 8.81% and 35.6% respectively. Lee *et al*. [16] combined a CL with DBN to scale down the large image and invariant to local translation. With the above-mentioned literature reviews, the effect of stacking FCN and CNN sequentially has not been fully studied instead of going deeper architecture.

In this paper, investigation of the effect of stacking three types of network layers sequentially, *i.e*. (1) CL; (2) PL and; (3) FL. The layers are sequentially added in the network to understand the effect of networks layers in term of classification accuracy. Subsequently, convolutional deep feedforward network (C-DFN) consisting of one CL, one PL and five FLs is proposed in the network. The CL is used as a trainable feature extractor and obtains features which are invariance to translations, rotation and scale before fed into the four layers of FCN which acts as universal approximator for classification. The main contributions of this paper are: (1) Investigation of the effect to stack three type network layers, *i.e*. CL in three filter size, PL and FL; (2) the proposed C-DFN consisting of one CL, one PL and four FLs for classification; (3) Four different types of architecture performance *i.e*. DFN, C-DFN, DBN, ad C-DBN were investigated on MNIST dataset, Daimler pedestrian dataset, and INRIA person dataset. This paper is organised as follows: Section 2 describes the architecture of C-FDN. Section 3 describes the experimental setup and experimental results discussion. Section 4 concludes this chapter.

2. CONVOLUTIONAL DEEP FEEDFORWARD NETWORK

Convolutional deep feedforward network (C-DFN) [17] is made up of a convolutional layer (CL), a pooling layer (PL), and followed by four fully connected layers (FLs) as shown in Fig. 1. CL is used as a trainable feature extractor to extract unique features from the input image which are invariant to scale, rotation, and translation [17]. In this section, the C-DFN architecture will be described in detail. The architecture process is divided into two parts: forward propagation and backpropagation computation.

2.1 Forward Propagation

The input data is forward propagated from input layer to multiple hidden layers until it reaches the output layer to produce an output signal. In the following section, the forward propagation computation is described in detail in each layer of the C-DFN.

2.1.1 Convolutional layer

In the convolutional layer, the input image, d_i is convoluted with $i \times j$ learnable kernels, k_{ij} and go through the activation function, $f(\cdot)$ to form the output feature map, v_j^l. The convolutional operation is defined as follows [18]:

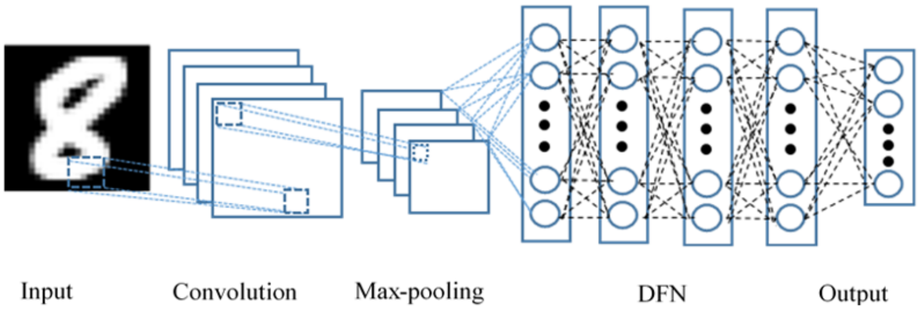

Fig. 1. Convolutional deep feedforward network.

$$v_j^l = f\left(\sum_{i \in M_j} d_i^{l-1} * k_{ij}^l + b_j^l\right), \tag{1}$$

where M_j represents input maps. d_i^{l-1} represents i^{th} input channel from the previous layer. v_j^l denotes output of convolutional layer. l represented the number of layer. Each output map is given an additive bias, b_j after the input image is convoluted with learnable kernels.

2.1.2 Max pooling layer

Once input image is convoluted with kernels, a subsampling layer is used to build up spatial and configural invariance. It could reduce the computing process by reducing half of the feature maps size. Max pooling is performed in non-overlapping neighbourhood using Eq. (2).

$$z_{ijk} = max(v_{i,j+n,k+m}). \tag{2}$$

Max pooling obtains the maximum value in the neighbourhood of $n \times m$. The output of the subsampling layer is subsequently arranged into an input vector of deep feedforward network (DFN).

2.1.3 Deep feedforward layer

Deep feedforward network is made up of multiple fully connected layer (FL). The input signal is multiplied with the weight connection, w_{ji} and then plus with a bias. A fully connected layer can be formed using Eq. (3).

$$x_j^l = \sum_{i=1}^{m} w_{ji} \times y_i^{l-1} + b_i, \tag{3}$$

where m is the total number of inputs applied to neuron j. y_i^{l-1} is input of FL from the previous layer's output. The output signal y_j^l of FL is given as follows:

$$y_j^l = f(x_j^l). \tag{4}$$

where $f(\cdot)$ is the activation function of the network. The output signal is then compared with the targeted output to produce an error back-propagated into the network layer by layer, which is described in the next section.

2.2 Backpropagation

In the backpropagation, an error signal is calculated by comparing the output of the network with the targeted output. The resulting error signal is propagated through the network in backward direction, layer by layer. In this paper, stochastic gradient descent backpropagation algorithm [19] is used to update the weights connection in C-DFN. After the parameters of DFN is updated, the error signal is backpropagated to the subsampling layer and convolutional layer. For the subsampling layer, the local gradient from the DFN is multiplied with the subsampling layer. The local gradients of convolutional layer for kernels update is defined as follows [18]:

$$\delta_j^l = f'(u_j) \cdot up(\delta_j^{l+1}), \tag{5}$$

where $f'(\cdot)$ is the derivatives of activation function, u_j is $\sum_{i \in M_j}(d_i^{l-1} * k_{ij}^l + b_j^l)$ as the pre-activation input, and $up(\cdot)$ represents up-sampling the local gradients from layer $l+1$, which is the local gradients from subsampling layer. The gradient of bias, b_j is summing all the entries of δ_j^l as follows:

$$\frac{\delta E}{\delta b_j} = \sum_{q,r} (\delta_j^l)_{qr}. \tag{6}$$

Finally, the gradients to updates the kernels are computed using Eq. (7).

$$\frac{\delta E}{\delta k_{i,j}^l} = \sum_{q,r} (\delta_j^l)_{qr} (p_i^{l-1})_{qr}, \tag{7}$$

where $(p_i^{l-1})_{qr}$ is the patch in v_i^{l-1} that is element-wise multiplication with $k_{i,j}^l$. It computes the element at (q,r) in the output convolutional map v_j^l.

3. SIMULATION RESULTS AND DISCUSSION

Four types of network structure were investigated, *i.e.* DBN, C-DBN, DFN and C-DFN. The experiments involved the change of filter size in a CL, stacking a number of CL, stacking a number of FL, and stacking a number of convolutional and pooling layer (CP). All the network structures were developed using MATLAB 2013b software and TitanX GPU. The convolutional kernel is initialized with Gabor filter [20]. Four Gabor filters [21, 22] is used as the convolution kernels in the CL to extract the pertinent information from the raw image 28×28. In the experiments, Leaky Rectified Linear Unit (LReLU) with $a = 0.25$ is set in the activation function of the network [23]. Besides that, weight decay 1×10^{-4} is implemented during the training process. All the experimental results were the average results of three times repeated experiment simulations for 100 training epochs. The network structures investigation were performed on handwritten digit dataset (MNIST). The architecture performance comparison between DFN, DBN, C-DFN, and C-DBN were conducted on MNIST dataset [3], INRIA person dataset [24], and Daimler classification dataset [25].

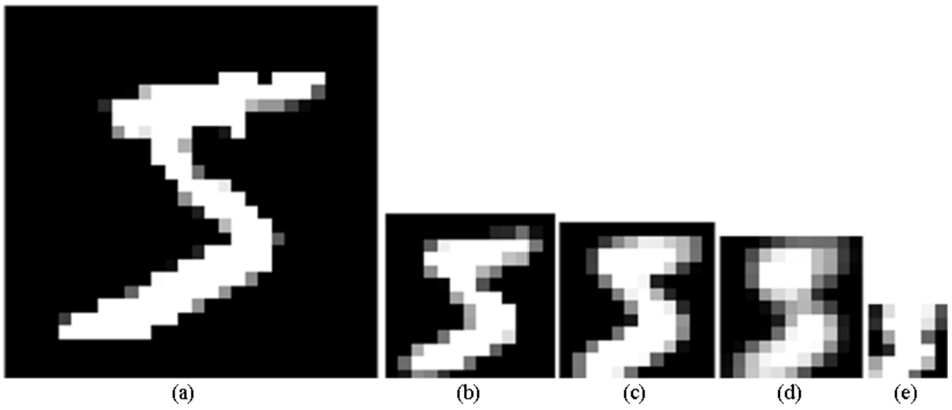

Fig. 2. Input for fully connected layer in different types of network structure (a) raw input (28×28); (b) (3×3) CP output; (c) (5×5) CP output; (d) (7×7) CP output; (e) (3×3) CPCP output.

Table 1. Misclassification rate of three type of Gabor filter sizes.

Network Structure	Train Misclassification	Testing Misclassification
(3×3) CPFL	11.94	11.86
(5×5) CPFL	14.00	13.35
(7×7) CPFL	16.51	17.66

3.1 Experiment #1: Change of Filter Size in the Network

In the first experiment, the relationship between filter sizes and the performance of the network was investigated to conclude the effect of filtering size in the network. CPFL network consists of one CL, one PL, and one FL. Table 1 showed the MNIST dataset experimental results. This experiment investigated three different sizes of Gabor filter used in CL *i.e.* 3×3, 5×5 and 7×7. Based on the experimental result, smaller filter size achieved the lowest misclassification rate which is 11.86% in MNIST dataset. This is because smaller filter size was able to capture very fine details of the image while having a bigger filter size will leave out minute details in the image. The effect of using different size of filter can be observed in Fig. 2 with blurry output image.

3.2 Experiment #2: Stacking Fully Connected Layers

As concluded from Section 3.1, the best filter size for convolutional layer was 3×3 because smaller filter size could retain more spatial information from the input. In the experiment #2, the filter size was fixed to 3×3. CP2FL was referred to one CL, one PL and two FLs. This experiment stacked up to five FLs sequentially to investigate the performance of the network. Table 2 showed that stacked up to four FLs obtained the lowest misclassification rate which is 2.6% in MNIST dataset. Stacking five layers of FLs did not reduce the testing misclassification rate. This was due to the excessive number of neurons in a network will lead to overfitting problem.

Table 2. Misclassification rate of increasing number of stacked fully connected layer.

Network Structure	Train Misclassification	Testing Misclassification
CPFL	11.94	11.86
CP2FL	1.15	4.29
CP3FL	0.63	2.83
CP4FL	0.76	2.6
CP5FL	0.7	2.86

Table 3. Misclassification rate of increasing number of stacked convolutional layer.

Network Structure	Train Misclassification	Testing Misclassification
CPFL	11.94	11.86
2CPFL	11.7	10.91
3CPFL	15.55	14.41

3.3 Experiment #3: Stacking Convolutional Layers

This experiment uses CPFL as a baseline to investigate the network performance by stacking multiple layers of the CL. 2CPFL was referred to two CLs, one PL and one FL. Table 3 showed that adding a convolutional layer on CPFL with the reduction effect of the testing misclassification rate from 11.86% to 10.91%. Adding one more convolutional layer on 2CPFL the misclassification rate increases to 14.41%. However, according to [7], stacking of two CLs with 3×3 filter size has the same effective receptive field of the 5×5 CL. Therefore, in this experiment, the training and testing performance of 2CPFL had achieved a better recognition rate than CPFL with 5×5 filter size. Similarly, 3CPFL had performed better than CPFL with 7×7 filter size as shown in Table 1.

3.4 Experiment #4: Stacking Convolutional Layers and Subsampling Layers

In experiment #4, CPFL was applied as a baseline to investigate the effect of adding PL after every CL of the network structure. Experimental results in Table 4 showed that adding more PL contributed to worse misclassification rate. The effect of PL was to reduce the dimensionality of feature maps while remaining the rotational and shift invariant features of the local region. The spatial relationship between local regions are disappear due to the region compression. Therefore, adding more PLs introduced the information loss for the classification task.

Table 4. Misclassification rate of three type of stacked CP layers.

Network Structure	Train Misclassification	Testing Misclassification
CPFL	11.94	11.86
CPCPFL	15.59	14.68
CPCP2FL	16.3	14.56
CPCPCPFL	68.81	69.97

3.5 Comparison of DFN, DBN, C-DFN, and C-DBN

After the experiments were being conducted in the previous section, CP4FL network structure consisting of one CL, one PL and four FLs obtained the lowest misclassification rate 2.6% in MNIST dataset. In this network, the relationship of inter-connected pixels is fully linked with weights parameter although the number of convolutional kernels is small. In addition, a stack of equal length of the FL has relatively improved the performance of the recognition rate when the network is going deeper.

In this experiment, CP4FL is renamed as C-DFN. This section compared DFN, C-DFN, DBN, and C-DBN architecture performance on MNIST dataset, Daimler pedestrian dataset, and INRIA person dataset. In addition, PReLU activation function is added to the networks for further investigation. In Table 5, C-DFN with LReLU activation function achieved the lowest misclassification rate among all the network architectures. Based on three datasets performance evaluation showed in Table 5, C-DFN with LReLU reduced misclassification rate to the average of 9.41%. C-DFN with PReLU reduced the misclas-

Table 5. Performance evaluation of FCN architectures on three dataset using LReLU and PReLU.

Dataset	Misclassification rate (%)							
	LReLU				PReLU			
	DFN	C-DFN	DBN	C-DBN	DFN	C-DFN	DBN	C-DBN
MNIST	1.73	1.32	1.74	1.48	1.74	1.59	1.60	1.47
INRIA	16.45	13.11	17.27	13.42	15.72	12.69	15.90	12.60
Daimler	13.85	13.81	15.42	15.47	13.49	13.39	15.16	15.31
Average	10.68	9.41	11.48	10.12	10.32	9.22	11.04	9.79

sification rate to the average of 9.22%. C-DBN with LReLU reduced misclassification rate to the average of 10.12%. C-DBN with PReLU reduced the misclassification rate to the average of 9.79%. Therefore, the CL acts as a trainable feature extractor improved the network performance significantly. Moreover, it reduced 14% of the trainable parameters in DFN.

4. CONCLUDING REMARKS

Three basic layers *i.e.* convolutional layer (CL), pooling layer (PL) and fully connected layer (FL) are sequentially added for performance investigation. In CL, the smallest size of the receptive field captures better local region of an input. However, if the size of the convolutional layer has not grown wider, the recognition rate may become poor due to the loss of connecting information. The requirement of increasing the number of filters across one layer is important to improve the connectivity of local information. PL has the effect of subsampling for reducing dimensionality and introducing some degree of invariance to local translation and distortion in the input. In a FL, the relationship of inter-connected pixels are linked with weight parameters. Therefore, the stack of equal-length of the FL has relatively improved the performance of the recognition rate when the network is grown deeper. In addition, experimental results showed that a CL with four FLs (C-DFN) performed significantly better than DFN, DBN, and C-DBN using three datasets. As a result, C-DFN with LReLU recorded an average misclassification of 9.41% as compared to C-DBN with average misclassification rate of 10.12%. On the other hand, C-DFN with PReLU achieved the lowest misclassification rate of 9.22% as compared to C-DBN with 9.79%. For future work, parametric activation function shall be further investigated to improve and adapt the network configuration in deeper network.

ACKNOWLEDGMENT

The authors gratefully acknowledge the support of NVIDIA Corporation for the donation of a Titan X and Quadro P6000 GPU in this research.

REFERENCES

1. A. Uçar, Y. Demir, and C. Güzeliş, “Moving towards in object recognition with deep learning for autonomous driving applications,” in *Proceedings of International Symposium on Innovations in Intelligent Systems and Applications*, 2016, pp. 1–5.
2. O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, “Convolutional neural networks for speech recognition,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, Vol. 22, 2014, pp. 1533–1545.
3. Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, Vol. 86, 1998, pp. 2278–2324.
4. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Proceedings of the 25th International Conference on Neural Information Processing Systems*, Vol. 1, 2012, pp. 1097–1105.

5. J. Deng, W. Dong, R. Socher, L. Li, K. Li, and F.-F. Li, "Imagenet: A large-scale hierarchical image database," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.
6. M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, Vol. abs/1311.2901, 2013.
7. K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, Vol. abs/1409.1556, 2014.
8. C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.
9. K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.
10. G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-normalizing neural networks," *CoRR*, Vol. abs/1706.02515, 2017.
11. G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, Vol. 18, 2006, pp. 1527–1554.
12. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, Vol. 15, 2014, pp. 1929–1958.
13. I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," in *Proceedings of the 30th International Conference on International Conference on Machine Learning*, Vol. 28, 2013, pp. III–1319–III–1327.
14. L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, Vol. 28, 2013, pp. 1058–1066.
15. M. Lin, Q. Chen, and S. Yan, "Network in network," *CoRR*, Vol. abs/1312.4400, 2013.
16. H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng, "Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations," in *Proceedings of the 26th ACM Annual International Conference on Machine Learning*, 2009, pp. 609–616.
17. M. M. Lau, J. T. S. Phang, and K. H. Lim, "Convolutional deep feedforward network for image classification," in *Proceedings of the 7th International Conference on Smart Computing & Communications*, 2019.
18. J. Bouvrie, "Notes on convolutional neural networks," 2006.
19. S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed., Prentice Hall PTR, 1998.
20. N. Petkov and M. Wieling, "Gabor filter for image processing and computer vision," http://matlabserver.cs.rug.nl/edgedetectionweb/index.html, 2008.
21. J. L. Chu and A. Krzyżak, "Analysis of feature maps selection in supervised learning using convolutional neural networks," in *Proceedings of Canadian Conference on Artificial Intelligence*, 2014, pp. 59–70.
22. S. Luan, C. Chen, B. Zhang, J. Han, and J. Liu, "Gabor convolutional networks," *IEEE Transactions on Image Processing*, Vol. 27, 2018, pp. 4357-4366.

23. M. M. Lau and K. H. Lim, "Investigation of activation functions in deep belief network," in *Proceedings of the 2nd International Conference on Control and Robotics Engineering*, 2017, pp. 201–206.
24. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 1, 2005, pp. 886–893.
25. S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 28, 2006, pp. 1863–1868.

Mian Mian Lau received the Bachelor degree in Electronic and Communication Engineering from Curtin University Malaysia, in 2014. She is currently a Ph.D. candidate in Electronic and Communication Engineering in Curtin University Malaysia. Her research interests lie in deep learning with application to computer vision particularly pedestrian detection system.

King Hann Lim received his Master of Engineering (M.Eng) and Ph.D. degree in Electrical and Electronic Engineering in 2007 and 2012 respectively. He is currently a staff member in the Department of Electrical and Computer Engineering at Curtin University Malaysia. His research area includes image/video processing, artificial intelligence, and intelligent transportation system. He is a IEEE senior member in 2017. He has published more than 60 journals and conference papers in the related areas of his research interest.