Ncfm: Accurate Handwritten Digits Recognition using Convolutional Neural Networks

[2]Yan Yin, [12]JunMin Wu, [2]HuanXin Zheng
University of Science and Technology of China
[1]Suzhou Institute for Advanced Study
[2]Department of Computer Science and Technology
SuZhou, China
yy1001@mail.ustc.edu.cn, jmwu@ustc.edu.cn, zhxfl@mail.ustc.edu.cn

***Abstract*—Convolutional Neural Networks(CNNs) have been confirmed as a powerful technique for classification of visual inputs like handwritten digits and faces recognition. Traditional convolutional layer's input feature maps are convolved with learnable kernel then combined for achieving better performance. The biggest drawback is that the combination of feature maps can lose features and do not apply well to large-scale neural networks. In this paper, we introduce Ncfm(No combination of feature maps), a novel technique to improve the performance of CNNs. By applying Ncfm technique, the input feature maps are not combined, which implies that the number of input feature maps is equal to output feature maps. The Ncfm technique converges faster and performs better than Cfm (Combination of feature maps) with fewer filters. Through the type of feature map, experimental evaluation shows that the performance is improved and we achieve the state-of-the-art performance with 99.81% accuracy rate on the MNIST datasets.**

Keywords—Ncfm; Cfm; CNNs; Handwritten Digits Recognition;

I. INTRODUCTION

Nowadays, convolutional neural networks have become a hot topic in machine learning community, showing significant gains over start-of-the-art machine learning methods used in various applications, like speech, image processing and sentence classification[15,16,17,18,19,20]. Each application requires higher accuracy performing closely to human by neural networks. More importantly, we need higher accuracy rate, even to one hundred percent, and faster training. To achieve these goals, some methods and tricks are proposed for CNNs. Deep big simple neural net without convolutional layers gets 0.35% error rate [12]. In order to avoid overfitting, elastic distortion [4] was introduced to expand dataset and their simple network results in 0.4% error rate. Multi-column deep neural networks [5] average many networks' prediction and get 0.23% error rate. Big networks with billions of parameters may easily overfit even with the largest datasets. Recently, Hitton etal.[7] propose a new form of regularization called Dropout, extensive experiments show that it significantly reduces overfitting and improves test performance. Li etal.[6] introduces Dropconnect which is a generalization of Dropout and results in 0.21% error rate.

However, we observe that these CNNs apply traditional combination of feature maps in the convolutional layer, which can result in information loss and slower convergence. Traditionally, the previous layer's output feature maps are convolved with current layer's learnable kernels and then the multiple input feature maps are combined[3]，which reduces the number of neurons and parameters of networks. But we find that traditional combination ignores some useful features and leads to worse performance when features are combined. And in feedback pass of CNNs with Cfm, there are some reduction operations per training step, which can lead to much synchronization overhead in the massively-parallel GPU architecture.

In this paper, we introduce a novel Ncfm (No combination of feature maps) technique to abstract a variety of features. Each input feature map can get a output feature map without combining, which can avoid information loss and get higher accuracy. Evaluation results show that performance can be improved and we achieve the state-of-the-art accuracy rate with 99.81% on the MNIST datasets.

Specifically we make the following contributions:

1. We propose CNNs with Ncfm technique to improve accuracy and speed up convergence. We reach the state-of-the-art performance with 99.81% accuracy rate on the MNIST dataset.

2. We compare the Cfm(Combination of feature maps) technique with Ncfm technique and analyze the advantages. Firstly, Ncfm converges faster than Cfm. Secondly, Ncfm performs better than Cfm with fewer convolutional filters.

This paper is organized as follows: Section II describes the traditional Cfm technique and then introduces Ncfm technique to improve the performance of the convolutional neural networks performance. Section III introduces Ncfm technique in detail, illustrates the main advantages. Section IV presents the architectures of the convolutional neural networks with Ncfm and Cfm, which are used throughout section V. Section V designs three experiments to confirm the validity of the Ncfm technique. Finally, Section VI presents our conclusions.

II. COMBINATION OF FEATURE MAPS

In this section, we describe the traditional Cfm technique and observe that it results in information loss, which is not

978-1-5090-0620-5/16/$31.00 ©2016 IEEE

applied well to handwritten digits recognition. Then we introduce Ncfm technique to improve the performance of convolutional neural networks.

Convolutional layers and max-pooling layers are at the heart of the LeNet [8] family models. There are many convolutional kernels in each layer, and each kernel is initialized differently. The function of convolution operator is to extract different features from the input image. Its feature description capacity varies as the number of layers increases. The first convolution layer obtains the low-level features, like edges, lines and corners. The more layers the network have, the higher-level features it gets. In general, we have:

$$x_j^l = f\left(\sum_{i \in M_j} x_i^{l-1} * k_j^l + b_j^l\right) \tag{1}$$

where M_j represents a selection of input feature maps from the previous layers' feature maps. Some common choices of input feature maps include all pairs or triplets. Each output feature map is given an additive bias b.

It is advantageous to provide an output feature map that involves a sum over several convolutions of different input feature maps. In the literature, the input feature maps are combined to form an output feature map typically chosen by hand. However, Jack Bouvrie[3] attempted to learn such combinations during training. Let a_{ij} denotes the weight given to input feature maps x_i^{l-1} when forming output feature maps x_i^l. Then output feature map is given by:

$$x_j^l = f\left(\sum_{i=1}^{N_{in}} a_{ij}\left(x_i^{l-1} * k_j^l\right) + b_j^l\right) \tag{2}$$

In order to encourage some of the weights to reach zero value, we can add a regularization penalty to the final error function:

$$E' = E + \lambda \sum_{i,j} \left|(a)_{ij}\right| \tag{3}$$

This method is better than combining the input feature maps directly, but there are information loss at the same time and the weight a_{ij} is not easy to be trained.

Fig.1 shows the flow process of convolutional neural network with Cfm technique, in which

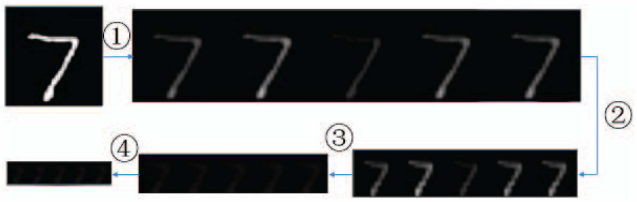
① represents the first convolutional layer with 5 filters.

② represents the first Max-pooling layer with size 2X2

③ represents the second convolutional layers with 5 filters.

④ represents second Max-pooling layer with size 2X2

Fig. 1. Flow process with combination of feature maps



Sometimes, feature maps combination can make outliers become robust, but it can result in losing feature when features of train examples are not complicated. As Fig.1 shows, we combine the input feature maps to represent more complicated information. We find that same features are lost after layer ③ due to the combination of feature maps, which reduces the performance.

Actually, we know fully-connection layers will combine previous layer's output to learn higher-level feature as well, so we do not combine input feature maps in the convolutional layers. The duty of convolutional layers is to extract more different feature maps without information loss to fully-connection layers.

As Fig.2 shows, the ③ and ④'s output feature maps still have more information to represent original input image than the Cfm technique. At the same time, the same number of filters produces more different output feature maps. The section III will discuss more details about the Ncfm technique.

Fig. 2. Flow process with no combination of feature maps

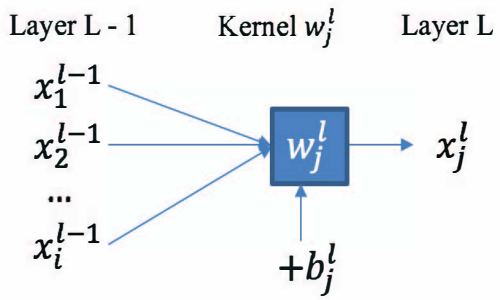
III. No Combinations of Feature Maps

In this section, we describe the details of the Ncfm technique. There are four parts. Part A describes the feedforward of convolutional layers applying Ncfm and part B describes the backpropagation. In the part C, we apply Ncfm to convolutional layers and analyze the main advantages.

A. Feedforward Pass in Convolutional Layer

We start describing the Ncfm technique by the simplest possible convolutional layer of neural network as illustrated below:

Fig. 3. A single neuron combines inputs



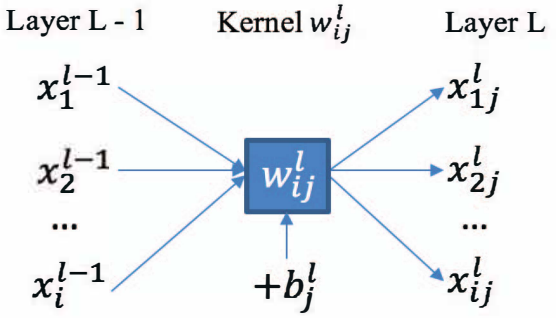
As Fig.3 shows, convolutional filter w_j^l is a computational unit that takes x_i^{l-1} and intercept term b^l as inputs, and we combine all feature maps, then produces the one output :

$$x_j^l = f(u_j^l),\ u_j^l = \left(W_j^l\right)^T X^{l-1} + b_j^l = \sum_i w_j^l x_i^{l-1} + b_j^l \qquad (4)$$

where $f : R \to R$ is called the activation function, w and b is the neuron's parameters.

If we do not combine the inputs, the following illustration shows the process:

Fig. 4. A single neuron does not combine inputs



As Fig.4 shows, filter w_{ij}^l convolves with each input x_i^{l-1} produces one output x_{ij}^l and the output is described:

$$x_{ij}^l = f(u_{ij}^l), u_{ij}^l = w_{ij}^l x_i^{l-1} + b_j^l \qquad (5)$$

The error loss function of convolution neural networks varies. Softmax regression is useful for classification as MNIST digits classification, where the goal is to distinguish among 10 different numerical digits. Now we describe the cost function that we'll use for softmax regression. In the equation blew, $1\{\bullet\}$ is the indicator function, so that $1\{$a true statement$\}=1$, and $1\{$a false statement$\}=0$.

$$E = \frac{1}{m}\left[\sum_{i=1}^{m}\sum_{j=1}^{k} 1\left\{y^{(i)} = j\right\} \log \frac{e^{\theta_j^T x^{(i)}}}{\sum_{l=1}^{k} e^{\theta_l^T x^{(i)}}}\right] \qquad (6)$$

where m is the number of train examples, k is the number of categories, θ is the parameter of neural networks, $x^{(i)}$ is the final feature, $y^{(i)}$ is the label.

B. Backpropagation Pass in Convolutional Layer

In order to minimize (6) as the function of the parameters W and b, we apply traditional optimization algorithm like batch gradient descent to train the neural network [13]. One iteration updates the parameters W and b as following:

$$w_{ij}^l = w_{ij}^l + \alpha \frac{\partial E}{\partial w_{ij}^l} \qquad (7)$$

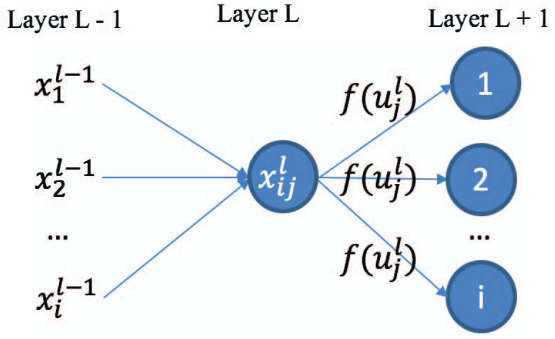
$$b_j^l = b_j^l + \alpha \frac{\partial E}{\partial b_j^l} \qquad (8)$$

where α is the learning rate.

According to (8), b_j^l is updated base on the value of $\frac{\partial E}{\partial b_j^l}$. We define δ_j^l to describe expression $\frac{\partial E}{\partial b_j^l}$ and get the following evolution:

$$\delta_j^l = \frac{\partial E}{\partial b_j^l} = \frac{\partial E}{\partial u_j^l}\frac{\partial u_j^l}{\partial b_j^l} = \frac{\partial E}{\partial u_j^l} \qquad (9)$$

Fig. 5. Neurons combine inputs and fully connect to next layer.



As Fig.5 shows, neuron x_{ij}^{l} fully connects to layer L + 1 and combines all inputs from Layer L – 1. we can get δ_j^l according to the higher layer's δ_j^{l+1} :

$$\delta_j^l = \frac{\partial E}{\partial u_j^l} = \sum_k \frac{\partial E}{\partial u_k^{l+1}} \frac{\partial u_k^{l+1}}{\partial u_j^l} = \sum_k w_{jk}^{l+1} \delta_k^{l+1} f'(u_j^l) \tag{10}$$

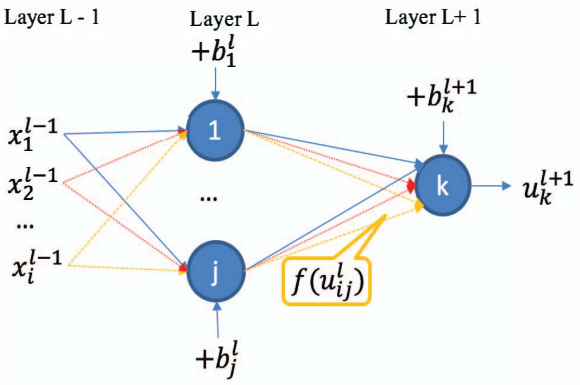
Based on (7) and Fig 5, in order to update w_{ij}^l, we should get the derivative of the loss function (6):

$$\frac{\partial E}{\partial w_{ij}^l} = \frac{\partial E}{\partial u_j^l} \frac{\partial u_j^l}{\partial w_{ij}^l} = \delta_j^l x_i^{l-1} \tag{11}$$

Where u_j^l is defined by (4).

When we do not combine the inputs, every neuron will get the same number of output units as input. As the Fig.6 shows, neuron j does not combine layer L - 1's inputs and fully connect to Layer L + 1.

Fig. 6. Neurons do not combine inputs and fully connect to next layer



In this case, we convert (9) to

$$\frac{\partial E}{\partial b_j^l} = \sum_i \frac{\partial E}{\partial u_{ij}^l} \frac{\partial u_{ij}^l}{\partial b_j^l} = \sum_i \frac{\partial E}{\partial u_{ij}^l} = \sum_i \delta_{ij}^l \tag{12}$$

and we convert (10) to

$$\delta_{ij}^l = \frac{\partial E}{\partial u_{ij}^l} = \sum_k \frac{\partial E}{\partial u_k^{l+1}} \frac{\partial u_k^{l+1}}{\partial u_{ij}^l} = \sum_k w_{jk}^{l+1} \delta_k^{l+1} f'(u_{ij}^l) \tag{13}$$

and we convert (11) to

$$\frac{\partial E}{\partial w_{ij}^l} = \sum_i \frac{\partial E}{\partial u_{ij}^l} \frac{\partial u_{ij}^l}{\partial w_{ij}^l} = \sum_i \delta_{ij}^l x_i^{l-1} \tag{14}$$

Based on (12), (13) and (14), we update b_j^l and w_{ij}^l .

C. Compare Ncfm with Cfm in Convolutional Network s

We now apply this approach to convolutional layers. Suppose that each convolutional layer has C_i convolutional filters, one input feature map through n convolutional layers gets $\prod_{i=1}^{n} c_i$ output feature maps with Ncfm as Fig.2 shows. Then we convert (1) to

$$x_{ij}^l = f\left(x_i^{l-1} * k_j^l + b_j^l\right) \tag{15}$$

where x_{ij}^l , $i \in M_j$ is the output feature maps produced by filter k_j^l .

There are two main advantages of Ncfm technique:

- ReLUs is an activity function $f(x) = max(x,0)$. Based on (11) or (14), we can find the learning will happen when x is greater than zero. We know every neuron has many inputs and it is dangerous to combine all of them directly. If we use the Ncfm technique, we can find that every input trains separately as Fig.6 shows. That means when only one input is greater than zero, the learning will happen. This property will contribute to the fast convergence.
- As Fig.2 shows, Ncfm does not combine the input feature maps, so one input feature map through n convolutional layers gets $\prod_{i=1}^{n} c_i$ output feature maps. Under the same condition, one input can get $\sum_{i=1}^{c} c_i$ output feature maps with Cfm. In other words, Ncfm extracts more features to the fully-connection layer, which means it needs fewer convolutional filters than Cfm.

IV. Overall Architecture

In this section, we present the architectures of the convolutional neural networks with Ncfm and Cfm. These architectures are used throughout section V.

Fig.7 describes the overall architecture of the convolutional neural network. We apply the Ncfm technique in the second convolutional layers.

Fig. 7. Apply Ncfm technique on convolutional layers.

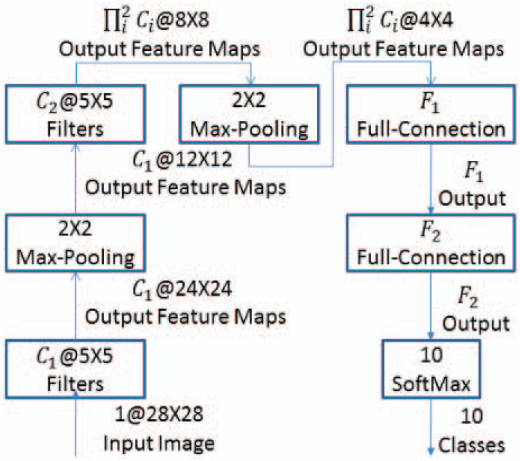
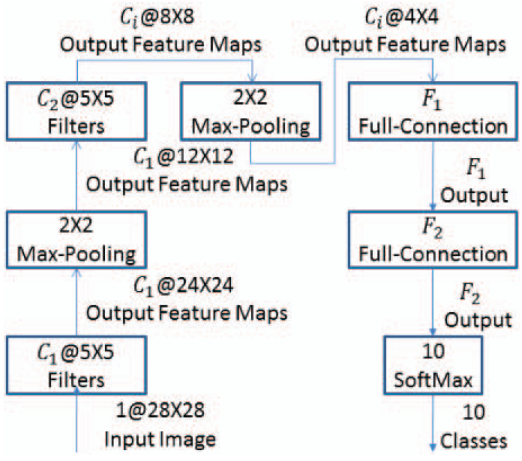


Fig. 8. Apply Cfm technique on convolutional layers.



In the first convolutional layer, there are C_1 filters with 5X5 convolutional kernel. The filters extract $C_1@24X24$ (C_1 feature maps with width 24 and height 24) feature maps from the input image $1@28X28$. After extracting features from convolutional layers, the Max-Pooling layer converts $C_1@24X24$ input feature maps into $C_1@12X12$ output feature maps.

The second convolutional layer with $C_2@5X5$ filters converts every input feature map from previous layer into $C_2@8X8$ output feature maps. That means every input feature map will get C_2 output feature maps. After the extraction of Max-Pooling layer, we get $\prod_i^2 C_i@4X4$ output feature maps.

The first fully-connection layer with F_1 neurons are connected with the previous layer's output features $4X4X\prod_i^2 C_i$, so its weight matrix is $(16\prod_i^2 C_i)X(F_1)$ and the output is a vector with length F_1. The second fully-connection layer is similar to the first one. Its weight matrix is $F_1 X F_2$ and the output is a vector with length F_2.

The last is Softmax layer with only 10 neurons and the weight matrix is $F_2 X10$. The final output of this layer is the probability of prediction index. The maximum one is the final prediction result.

Fig.8 describes the architecture of network with Cfm as follow. In order to compare with Ncfm, Cfm's configuration is so similar to Ncfm that the only difference we can find is the second convolutional layer and Max-Pooling layer's output size.

V. Experiments

In this section, we design three experiments to confirm the validity of the Ncfm technique. Part A and part B describe the experimental setup like dataset and weight initialization. Part C is an experiment to confirm the fast convergence and Part D confirms that the Ncfm technique performs better than Cfm with fewer filters. Finally, Part E shows the best performance result of the MNIST dataset.

A. Dataset and Data Augmentation

We use the MNIST as dataset for our experiments. The MNIST [1] is a benchmark dataset of images of segmented handwritten digits and each image is 28X28 pixels. There are 60,000 training examples and 10,000 testing examples.

The commonest method to reduce overfitting on image data is artificially enlarging the dataset using label-preserving transformation [4,5,10].

We employ several distinct forms of data augmentation:

- Elastic Distortions [4]

The new target location is given with respect to the previous position:

$$(x,y)^{T\,arg\,et} = (x,y)^{Ori} + (\Delta x, \Delta y) \tag{16}$$

Where $|\Delta x| \le 1.0$ and $|\Delta y| \le 1.0$. We can initialize matrices Δx and Δy randomly and then reduce noise by using the Gaussian filter.

- Affine Transformation

We use rotation and scale transformation to enlarge the dataset. The rotation range is from $-12°$ to $12°$ and the scale range is from -12% to 12% of origin pixel size.

B. Initialization of The Nerual Network's Weight

All experiments use mini-batch gradient descent with momentum[11]. The weights of layers are initialized with normal distribution or uniform distribution.

- Convolutional layers

$$X \sim Normal(0,0.1) \tag{17}$$

- Fully-Connection layers

$$X \sim Uniform[-b,b], b = \sqrt{6.0/\sqrt{width + height}} \tag{18}$$

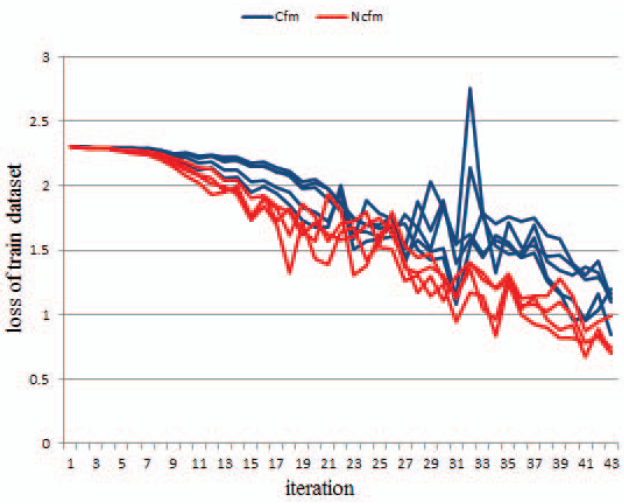
- Softmax layer

$$X \sim Uniform[-0.01,0.01] \tag{19}$$

C. Fast Convergence

Backpropagation is a very popular neural network learning algorithm because it is conceptually simple, computationally efficient, and it often works. It is important to obtain fast convergence for the back-propagation learning. As fig.2 shows, Ncfm can get more features, which can detemine the direction of the extremum more accurately, so as to achieve faster convergence.

Fig. 9. Convergence speed of Ncfm and Cfm for multiple experiments.



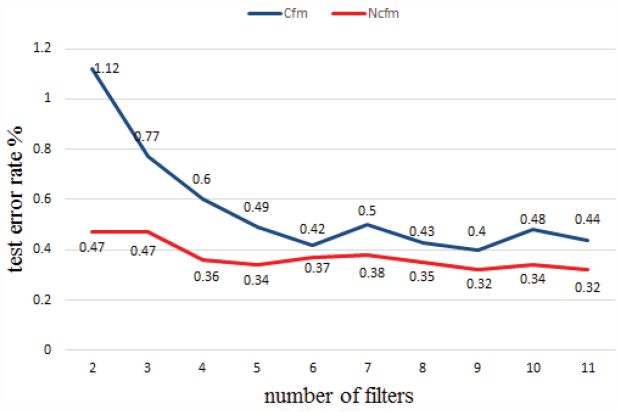
To further analyze the fast convergence property, we apply the Ncfm technique to the second convolutional layer and compare it with Cfm. The network configure is $C_1 = 10$, $C_2 = 20$, $F_1 = 256$, $F_2 = 256$. As Fig.7 and Fig.8 show, the first fully-connection layers' weight scale are different. And we consider the loss function that showed the equation (6).

Fig.9 shows value of the loss function as the iteration varies. We can find the network with Ncfm converges faster than Cfm.

D. Fewer Convolutional Filters Needed

The Ncfm technique extracts more features than the tradetional Cfm technique, so we can get better performance with fewer convolutional filters. The network's configuration is $C_1 = 10$, $F_1 = 256$ and varies C2 from 2 to 11. As Fig.10 shows, we can find the network performs with great consistency by using Ncfm but Cfm's error rate increases when the number of filters descends.

Fig. 10. Fewer convolutional filters needed.



E. Best Performance Result

In order to compare the Ncfm technique with traditional Cfm technique, we combine input feature maps directly in the second convolutional layer as Fig.6 shows.

The configuration for Ncfm is $C_1 = 10$, $C_2 = 20$, $F_1 = 256$, $F_2 = 256$. In order to further confirm the validity of the Ncfm technique, the configuration for Cfm is $C_1 = 10$, $C_2 = 200$, $F_1 = 256$, $F_2 = 256$. This configuration is fairer because the convolutional layer extract same number of feature maps to fully-connection layer. Finally, we get the following result.

TABLE I shows that without local response normalization [9] or enforcing sparse combinations [3] technique, Ncfm still performs better than Cfm.

TABLE I. COMPARE NCFM WITH CFM(ERROR RATE %)

Trial	Ncfm	Cfm
1	***0.24***	0.43
2	0.32	0.33
3	0.27	0.37
4	0.28	0.36
5	0.27	0.36
Avg.	0.276 ± 0.0258	0.37 ± 0.0329
Vote	***0.19***	0.29

The previous state-of-the-art is 0.28% error rate for a single model and 0.21% error rate with voting [6], which uses both local response normalization and enforcing sparse combinations technique. We note that Ncfm surpasses the state-of-the-art result, achieving 0.24% error rate for a single model and 0.19% error rate with voting.

VI. Conclusion

We propose Ncfm(No combination of feature maps) technique be applied to handwritten digits recognition. This technique can accelerate convergence because every input learns absolutely and performs better with fewer convolutional filters. With Ncfm technique we demonstrate state-of-the-art accuracy with 99.81% on MNIST datasets .

Ncfm is a novel technique that applied to convolutional layers to represent the higher abstract information and avoid the information loss. The experiments have proved that Ncfm is very suitable to handwritten digits recognition.

In the future, we will apply the Ncfm technique on more datasets and we may have some changes and tricks because of different input features.

REFERENCES

[1] Y.LeCun, "The MNIST database of handwritten digits," http://yann.lecun.com/exdb/mnist.

[2] LeCun Y, Boser B, Denker J S, et al. Backpropagation applied to handwritten zip code recognition[J]. Neural computation, 1989, 1(4): 541-551.

[3] Bouvrie J. Notes on convolutional neural networks[J]. 2006.

[4] Simard P Y, Steinkraus D, Platt J C. Best practices for convolutional neural networks applied to visual document analysis[C]//2013 12th International Conference on Document Analysis and Recognition. IEEE Computer Society, 2003, 2: 958-958.

[5] Ciresan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification[C]//Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on. IEEE, 2012: 3642-3649.

[6] Wan L, Zeiler M, Zhang S, et al. Regularization of neural networks using dropconnect[C]//Proceedings of the 30th International Conference on Machine Learning (ICML-13). 2013: 1058-1066.

[7] Hinton G E, Srivastava N, Krizhevsky A, et al. Improving neural networks by preventing co-adaptation of feature detectors[J]. arXiv preprint arXiv:1207.0580, 2012.

[8] Y.LeCun, "LeNet-5, convolutional neural networks," http://yann.lecun.com/exdb/lenet/

[9] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks[C]//Advances in neural information processing systems. 2012: 1097-1105.

[10] Cireşan D C, Meier U, Masci J, et al. High-performance neural networks for visual object classification[J]. arXiv preprint arXiv:1102.0183, 2011.

[11] LeCun Y A, Bottou L, Orr G B, et al. Efficient backprop[M]//Neural networks: Tricks of the trade. Springer Berlin Heidelberg, 2012: 9-48.

[12] Claudiu Ciresan D, Meier U, Gambardella L M, et al. Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition[J]. arXiv preprint arXiv:1003.0358, 2010.

[13] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, Caroline Suen, "UFLDL Tutorial," http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

[14] Dan Ciresan ,Ueli merier and Jvrgen Schmidhuber. Multi-column Deep Nuural Networks for Image Classification[J]. Technical Report No. IDSIA-04-12.

[15] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *CoRR*, abs/1312.4400, 2013

[16] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Computer Vision and Pattern Recognition, 2014. CVPR 2014. IEEE Conference on*, 2014.

[17] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition, 2014. CVPR 2014. IEEE Conference on*, 2014.

[18] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[19] K. Simonyan and A. Zisserman. Very deep convolu- tional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[20] N. Kalchbrenner, E. Grefenstette, P. Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. *In Proceedings of ACL 2014.*