Very Deep Neural Network for Handwritten Digit Recognition

Yang Li(✉), Hang Li, Yulong Xu, Jiabao Wang, and Yafei Zhang

College of Command Information Systems, PLA University of Science and Technology, Nanjing 210007, China
solarleeon@outlook.com

Abstract. Handwritten digit recognition is an important but challenging task. However, how to build an efficient artificial neural network architecture that can match human performance on the task of recognition of handwritten digit is still a difficult problem. In this paper, we proposed a new very deep neural network architecture for handwritten digit recognition. What is remarkable is that we did not depart from the classical convolutional neural networks architecture, but pushed it to the limit by substantially increasing the depth. By a carefully crafted design, we proposed two different basic building block and increase the depth of the network while keeping the computational budget constant. On the very competitive MNIST handwriting benchmark, our method achieve the best error rate ever reported on the original dataset (0.47 %±0.05 %), without data distortion or model combination, demonstrating the superiority of our work.

Keywords: Neural network · Convolutional neural networks · Deep learning · Handwritten digit recognition

1 Introduction

Handwritten digit recognition is a promising subfield of object recognition with various applications. In the last ten years, automatic handwritten digit recognition capabilities have dramatically improved due to advances in deep learning and convolutional neural networks (CNNs). The performance of the new methods on the well-known MNIST dataset have reduce the recognition error rate from several percentage points to 1 % [1], then down to 0.5 % [2], then down to 0.23 % [3].

However, almost all of the successful methods were trying to improve recognition accuracy in three different ways. The first method attempts to design a better architecture, which is better for handwritten digit recognition [4,5]. The second method improves the accuracy by enlarging the MNIST dataset. So far, the best results on MNIST were obtained by deforming training images, thus greatly increasing their number [3]. The third method combines several training models and makes decision by swarm intelligence [6].

H. Yin et al. (Eds.): IDEAL 2016, LNCS 9937, pp. 174–182, 2016.
DOI: 10.1007/978-3-319-46257-8_19

To the best of our knowledge, data augmentation and model combination can improve almost all of the recognition methods, but they also lead to more training time. Even if one was able to train many different large networks, using them all at test time would be infeasible in applications where it is important to respond quickly. One discouraging news is that a lot of this progress is not a consequence of new ideas, algorithms and improved network architectures, but mainly just the result of a larger dataset and combination of several models.

In this paper, we will focus on a new efficient deep neural network architecture for handwritten digit recognition. The basic idea of this paper is taking inspiration and guidance from the theoretical work by Andrew Zisserman et al. [7] and Christian Szegedy et al. [8], who use smaller receptive window size and smaller stride of convolutional layer to build very deep CNNs for ILSVRC-2014. By a carefully crafted design, we increased the depth of the network while keeping the computational budget constant. It was demonstrated that the representation depth is beneficial for the classification accuracy, and that the state-of-the-art performance on the MNIST dataset can be achieved using a CNNs architecture with substantially increased depth. The benefits of the architecture are experimentally verified on the MNIST dataset without data augmentation and model combination, where it could reach comparable performance of the state-of-the-art approaches with less computation burden and shorter training time.

The rest of this paper is organized as follows. Section 2 describes the proposed architecture using CNNs module in details. Then experimental results and comparisons are shown in Sect. 3. Finally, the conclusion and future work are given in Sect. 4.

2 The Proposed Architecture

In this section, we elaborate how to build a hierarchical feature extraction and classification system with CNNs module. As illustrated in Fig. 1, the model architecture is mainly based on CNNs. We first briefly review CNNs, and then we depict the proposed mode in details.

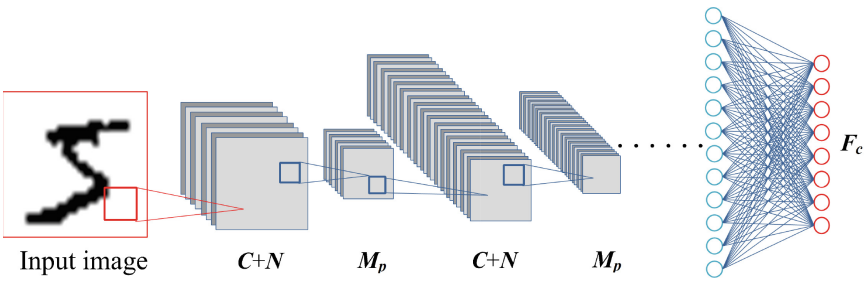

Fig. 1. CNNs feature extractor architecture.

2.1 A Brief Review of CNNs

Starting from LeCun [1], CNNs can be considered to be made up of two main parts. The first part typically had a standard structure stacked convolutional layers, which followed by contrast rectification layers (denoted as R) and average-pooling layers (denoted as A_p). The input of each layer is just the output of its previous layer. As a result, this forms a hierarchical feature extractor that maps the original input images into feature vectors. The second part are one or more fully-connected layers, which is a typical feed forward neural network trying to classify the extracted features vectors. In this paper, the proposed architecture is composed by 6 different component: convolutional layer (denoted as C), normalization layer (denoted as N), max-pooling layer (denoted as M_p), fully-connected layers (denoted as F_c), and we also adopt dropout (denoted as D_p) and Rectied Linear Units ($ReLU$) method in our network.

Convolutional Layer: In convolutional layer, each neuron is connected locally to its inputs of the previous layer, which functions like a 2D convolution with certain filter, then its activation could be computed as the result of a nonlinear transformation. In this paper, convolutional layer computes the convolution of the input image $x \in \mathrm{R}^{H \times W \times D}$ with a filter bank $f \in \mathrm{R}^{H' \times W' \times D \times D''}$ with D'' multi-dimensional. Formally, the output is $y \in \mathrm{R}^{H'' \times W'' \times D''}$ given by:

$$y_{i''j''d''} = b_{d''} + \sum_{i'=1}^{H'} \sum_{j'=1}^{W'} \sum_{d=1}^{D} f_{i'j'd} \times x_{i'+i''-1,j'+j''-1,d',d''} \tag{1}$$

where H represents the input image height, W represents the input image width, D represents the number of channel, H'represents filter height, W' represents filter width, D'' represents the number of filter bank, H'' represents the output image height, W'' represents the output image width.

Normalization Layer: The local contrast normalization layer is inspired by computational neuroscience models. Normalization layer applied independently at each spatial location and groups of channels to get:

$$y_{ijk} = x_{ijk}(\kappa + \alpha \sum_{t \in G(k)} x_{ijt}^2)^{-\beta} \tag{2}$$

For each output channel k, $G(k) \subset \{1, 2, ..., D\}$ is a corresponding subset of input channels. And the input image x and output image y have the same dimensions.

Pooling Layer: Pooling layer always produces down sampled versions of the input maps. This pooling involves executing some operation, typically average or max, over the activations within a small spatial region of each map of activations. Typically max pooling is preferred as it avoids cancellation of negative elements and prevents blurring of the activations and gradients throughout the network since the gradient is placed in a single location during back propagation. The max pooling operator computes the maximum response of each feature channel in a $H' \times W'$ patch, resulting in an output of size $y \in \mathrm{R}^{H'' \times W'' \times D}$:

$$y_{i''j''d} = \max_{1 \le i' \le H', 1 \le j' \le W'} x_{i'+i''-1, j'+j''-1, d} \tag{3}$$

ReLU: Typically the convolutional responses are passed through a non-linear activation function such as sigmoids, tanh, or $ReLU$ [14] to produce activation maps. The $ReLU$ can be compute as follows:

$$y_{ijd} = \max(0, x_{ijd}) \tag{4}$$

Dropout: The key idea of dropout is to randomly drop units (along with their connections) from the neural network during training. This method significantly reduces overfitting and gives major improvements over other regularization methods. The choice of which units to drop is random. In the simplest case, each unit is retained with a fixed probability p independent of other units, where p can be chosen using a validation set or can simply be set at 0.5, which seems to be close to optimal for a wide range of networks and tasks. For any layer l, $r^{(l)}$ is a vector of independent Bernoulli random variables each of which has probability p of being 1. This vector is sampled and multiplied element wise with the outputs of that layer $y^{(l)}$, to create the thinned outputs $\widetilde{y}^{(l)}$ [4].

$$r^{(l)} = Bernoulli(p) \tag{5}$$

$$\tilde{y}^{(l)} = r^{(l)} * y^{(l)} \tag{6}$$

Fully-Connected Layer: After multiple convolutional and pooling layers, a convolutional network typically has one or more fully connected neural net layers with weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ before the final classier. The entire network is trained with back-propagation of a supervised loss such as the cross entropy of a softmax classier output and the target labels y represented as a 1 of c vector, where c is the number of classes to discriminate.

$$y = -\sum_{ij} (x_{ijc} - \log \sum_{d=1}^{D} e^{x_{ijd}}) \tag{7}$$

2.2 Combining Modules into a Hierarchy Architecture

In order to design a new deep convolutional neural network architecture which is discriminative enough for handwritten digit recognition. Here we use the basic module to build up our model. However, different architectures can be produced by cascading the above-mentioned modules in various ways. According to [18], they point out the basic building block of convolutional networks are $C + A_p$ layer, $C + R + A_p$ layer, $C + R + N + A_p$ layer, $C + M_p$ layer. In this paper, we propose two different basic building block for our architecture.

***C+N+M_p* Layer:** This is the first basic building block of our convolutional networks. This block is composed of convolutional layer, normalization layer and max pooling down sampling layer. This block is just like human visual cortex

which is used for feature extraction and nonlinear dimensionality reduction in our model.

***C+N+ReLU* Layer:** This is second the basic building block of our convolutional networks, which compose of a convolutional layer followed by a normalization layer and a *ReLU* layer. This block is used for higher level feature extraction.

Table 1 shows the whole setting of our CNNs architecture. The input to our network is a fixed-size 28×28 gray image. The image is passed through a stack of different layers. Except for the first convolution filter, we use very small 3×3 receptive fields throughout the whole net. The max pooling layer here is non-overlapping and no rectification. Max-pooling is carried out over a 2×2 pixel window, with stride 2. It should be noticed that all weight convolutional layers are equipped with normalization. To the best of our knowledge, our CNNs architecture is the deepest model for handwritten digit recognition.

Table 1. CNNs architecture and parameters.

Layer	Type	Output size	Kernel size/Stride
1	Convolutional	$20 \times 24 \times 24$	$5 \times 5/1$
2	Normalization	$20 \times 24 \times 24$	—
3	Max pooling	$20 \times 12 \times 12$	$2 \times 2/2$
4	Convolutional	$40 \times 10 \times 10$	$3 \times 3/1$
5	Normalization	$40 \times 10 \times 10$	—
6	Max pooling	$40 \times 5 \times 5$	$2 \times 2/2$
7	Convolutional	$150 \times 3 \times 3$	$3 \times 3/1$
8	Normalization	$150 \times 3 \times 3$	—
9	ReLU	$150 \times 3 \times 3$	—
10	Convolutional	$150 \times 1 \times 1$	$3 \times 3/1$
11	Normalization	$150 \times 1 \times 1$	—
12	ReLU	$150 \times 1 \times 1$	—
13	Dropout(rate 0.4)	$150 \times 1 \times 1$	—
14	Convolutional	$150 \times 1 \times 1$	$1 \times 1/1$
15	Normalization	$150 \times 1 \times 1$	—
16	ReLU	$150 \times 1 \times 1$	—
17	Dropout (rate 0.1)	$150 \times 1 \times 1$	—
18	Fully connected	10	—

3 Experiments and Analysis

The experiments were run on the MNIST dataset. The MNIST dataset consists of handwritten digits 0–9 which are gray scale 28×28 pixel digit images. There are 60,000 training images and 10,000 testing images in total.

So far, the best results on MNIST were obtained by deforming training images, thus greatly increasing their number. This allows for training networks with many weights, making them insensitive to in-class variability. However, in this paper our network is not trained on numerous slightly deformed images, because we want to build a better learning model but not a simple model with seeing more data.

Our implementation is derived from the publicly available MatConvNet toolbox [9]. Matlab 2015a is used to conduct all the operations, running on a system with Intel Core i5-4690 CPU (3.50 GHz), 16 GB DDR3. Initial weights of the CNNs are drawn from a uniform random distribution in the range $[-0.01, 0.01]$. The training is carried out using mini-batch gradient descent (based on back-propagation) with momentum. The batch size was set to 100, momentum to 0.9. The training was regularized by weight decay (the L_2 penalty multiplier set to 5×10^{-4}) and dropout regularization for the two fully-connected layers (dropout ratio set to 0.4 and 0.1). The learning rate was initially set to 0.3, and then decreased by a factor of 3 when the validation set accuracy stopped improving.

The classification performance is evaluated using two measures: the top-1 and top-5 error. The former is a multi-class classification error, i.e. the ratio of incorrectly classified images; the latter is the main evaluation criterion used in the ILSVRC, and is computed as the ratio of images such that the ground-truth category is not within the top-5 categories. Figure 2 show one result of the training and testing accuracy on each epoch. From the left picture, we can see that the energy (training and testing loss) was decay dramatically with the training epoch. From the right picture, we can see that our method was learning fast which can get optimal performance 0.47 % error rate after 29 epoch iteration.

Finally, we compare our best-performing single network result with ten state-of-the-art methods. Table 2 shows the comparison of the recognition rate between the proposed architecture method and other recently reported results. All the methods in the experiment do not using data augmentation and model combination, our model secure the first place with $0.47\,\% \pm 0.05\,\%$ test error rate. To the best of our knowledge, this is the best error rate ever reported on the original MNIST dataset, without distortions or model combination. The best previously reported error rate was 0.53 % [18]. In addition, without data augmentation, our method dramatically relieve the training procedure. In terms of training time, the proposed architecture method takes 34.84 min. It is much faster than [3], which needs to train up to 35 CNNs and costs 14 h even when the GPU parallelization is carried out.

To further understand the learnt model, we also draw the first convolutional layer of the learnt filters in Fig. 3. An intriguing pattern is observed in the filters of MNIST dataset. We can see both horizontal and vertical stripes, for these patterns attempt to capture the edges of the images.

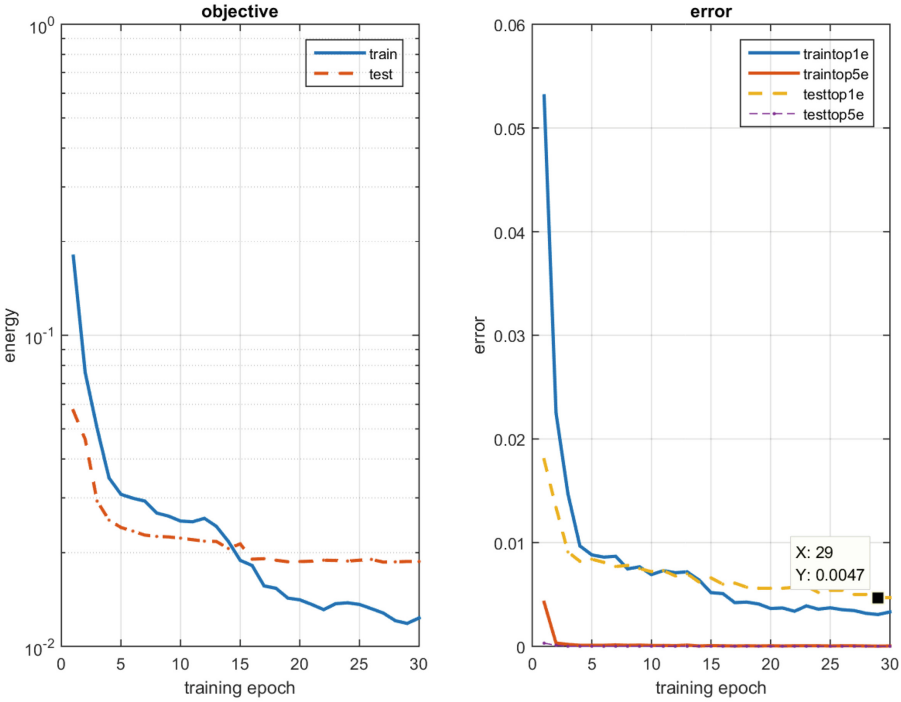

Fig. 2. Experimental results on MNIST dataset

Table 2. Test set misclassification rates for the best single model methods on the permutation invariant MNIST dataset.

Method	Test error
Srivastava et al. [4]	1.05 %
Salakhutdinov et al. [10]	0.95 %
Ranzato et al. [11]	0.60 %
Maxout NET [12]	0.94 %
Goodfellow et al. [13]	0.91 %
Deng et al. [14]	0.83 %
Rifai et al. [15]	0.81 %
Hinton et al. [16]	0.79 %
Zeiler et al. [17]	0.59 %
Jarrett et al. [18]	0.53 %
Our method	**0.42**% (0.47 % ± 0.05 %)

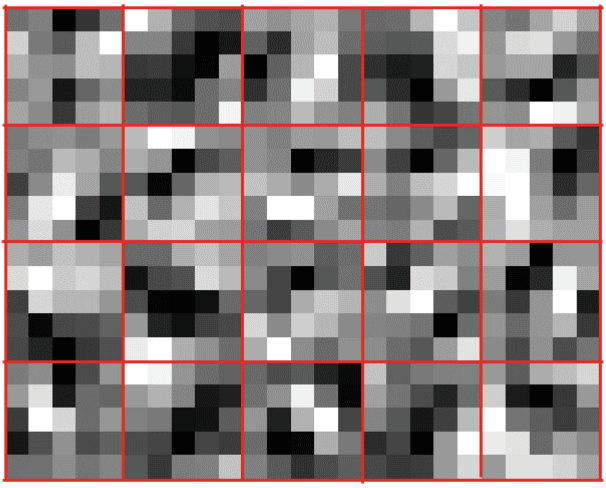

Fig. 3. The filters learned on MNIST dataset. There are 20 filter in the first stage and their size are 5×5

4 Conclusion and Future Work

In this paper, we propose a novel very deep network for handwritten digit recognition tasks. Unlike the shallow neural network used in many 1990s applications, ours are very deep. By a carefully crafted design, we propose two different basic building blocks and increase the depth of the network. The experiment result demonstrated that the representation depth is beneficial for the classification accuracy. Last and most importantly, the proposed model is a simple and computationally efficient approach for handwritten digit recognition tasks. On the very competitive MNIST handwriting benchmark, the proposed method achieve the best error rate ever reported on the original dataset, without distortions or model combination ($0.47\,\% \pm 0.05\,\%$). In the future, we will further explore the potential representation ability of CNNs for various visual recognition tasks.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2323 (1998)
2. Simard, P.Y., Steinkraus, D., Platt, J.C.: Best practices for convolutional neural networks applied to visual document analysis. In: 7th IEEE International Conference on Document Analysis and Recognition, pp. 958–963 (2003)
3. Ciresan, D., Meier, U., Schmidhuber, J.: Multi-column deep neural networks for image classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3642–3649 (2012)
4. Srivastava, N.: Improving neural networks with dropout. University of Toronto (2013)

5. Lin, M., Chen, Q., Yan, S.: Network in network. ArXiv preprint (2014). arXiv:1312.4400
6. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J., Cire, D.C., Meier, U., Gambardella, L.M.: Handwritten digit recognition with a committee of deep neural nets on gpus. ArXiv preprint (2011). arXiv:1103.4487
7. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (ICLR), pp. 1–14 (2015)
8. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–9 (2015)
9. Vedaldi, A., Lenc, K.: MatConvNet. In: 23th ACM International Conference on Multimedia, pp. 689–692 (2015)
10. Salakhutdinov, R., Hinton, G.E.: Deep Boltzmann machines. In: 12th International Conference on Artificial Intelligence and Statistics, pp. 448–455 (2009)
11. Ranzato, M.A., Poultney, C., Chopra, S., Lecun, Y.: Efficient learning of sparse representations with an energy-based model. In: Advances in Neural Information Processing Systems (NIPS), pp. 1137–1134 (2006)
12. Goodfellow, I.J., Warde-Farley, D., Mirza, M., Courville, A., Bengio, Y.: Maxout networks. ArXiv preprint (2013). arXiv:1302.4389
13. Goodfellow, I., Courville, A., Bengio, Y.: Joint training of deep Boltzmann machines for classification. In: International Conference on Learning Representations Workshops (ICLRW) (2013)
14. Deng, L., Yu, D.: Deep convex net: a scalable architecture for speech pattern classification. In: Proceedings of the Annual Conference of the International Speech Communication Association (INTERSPEECH), pp. 2285–2288 (2011)
15. Rifai, S., Dauphin, Y.: The manifold tangent classifier. In: Advances in Neural Information Processing Systems (NIPS), pp. 2294–2302 (2011)
16. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. ArXiv preprint (2012). arXiv:1207.0580
17. Wan, L., Zeiler, M., Zhang, S., LeCun, Y., Fergus, R.: Regularization of neural networks using dropconnect. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 1058–1066 (2013)
18. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), pp. 2146–2153 (2009)