Effects of Model Complexity on Generalization Performance of Convolutional Neural Networks

Tae-Jun Kim[1], Dongsu Zhang[2], and Joon Shik Kim[3]

[1]Seoul National University, Seoul 151-742, Korea, E-mail: tjkim@bi.snu.ac.kr
[2]Yangjae High School, Seoul, Korea, E-mail: 96lives@gmail.com
[3]University of Seoul, Seoul 130-743, Korea, E-mail: jskim.ozmagi@gmail.com

Abstract. Convolutional neural networks are known to be effective in learning complex image classification tasks. However, how to design the architecture or complexity of the network structure requires a more quantitative analysis of the architecture design. In this paper, we study the effect of model complexity on generalization capability of the convolutional neural networks on large-scale, real-life digit recognition data. We used the digit images of the MNIST dataset to train the neural networks and evaluated their performance on a test set of unobserved images. Using the LeNet software tool we varied the number of hidden layers and the number of units in the layers to evaluate the effect of model complexity on the generalization capability of the convolutional neural networks. In our experimental settings, we observe robust generalization performances of the convolutional neural networks on a wide range of model complexities. We analyze and discuss how the convolution layer and the sub-sampling layer may contribute to the generalization performance.

Keywords: Convolutional neural networks, MNIST, LeNet, image classification, model complexity, Occam's razor, generalization performance

1. Introduction

Computers solve many problems well if programmed appropriately by human programmers. However, some artificial intelligence problems, such as image and speech analysis, are hard to program the computers to solve. In this case, machine learning offers new possibilities since it allows to automatically build a program from data, i.e. by repeatedly observing humans solving the problem. An important issue in machine learning is how to control the complexity of the model: if the model is too simple, it cannot learn the data, whereas too complex models may overfit the data.

Convolutional neural networks are especially interesting as a machine learning model since they can learn complex patterns in real-life data sets, such as images. However, the design of the network structure, i.e. the number of layers and the number of units in the layers, remain an art. In this paper we aim to understand the relationship between the complexity of the network model and its generalization performance by exploring the architecture space experimentally. Our vision is to build a deep neural network that can learn to recognize human faces as more human faces are observed. In this first stage of our research, in this paper we experiment with the MNIST benchmark data sets.

The paper is organized as follows. In Section 2 we describe architecture and learning method of the convolutional neural network. In Section 3 we describe the data set and our experimental designs. Section 4 reports on the experimental results and their analysis. Section 5 concludes the work.

2. Deep Convolutional Neural Networks

Neural networks are neurobiologically inspired computational models of learning and memory. A single neuron *j* processes information in two stages: It first computes the net input net_j from the incoming activations x_i of presynaptic neurons $i \in \mathrm{Pre}$

$$net_j = \sum_{i \in \mathrm{Pre}}^{I} w_{ji} x_i$$

and then transfers the net input through a nonlinear activation function $\sigma(\cdot)$, such as a sigmoid function

$$\sigma(net) = \frac{1}{1 + \exp(-net)}.$$

The neurons are organized typically in a layered structure where a layer of neurons are connected from a previous layer of neurons and there are no connections between the neurons in the same layer. One of popular neural architectures is the multilayer perceptron consisting of two fully-connected feedforward layers of neurons. If *I* and *H* denote the numbers of input neurons and hidden neurons, respectively, the *k*-th output of the multilayer perceptron is expressed as

$$f_k(\mathbf{x}, \mathbf{w}) = \sigma_k\left(\sum_{h=1}^{H} w_{kh}^{(2)} \sigma_h\left(\sum_{i=1}^{I} w_{hi}^{(1)} x_i\right)\right)$$

where $\mathbf{x}$ is the input training pattern and $\mathbf{w} = (\mathbf{w}^{(1)}, \mathbf{w}^{(2)})$ is the weight vector that determines the neural network function. One of the most interesting features of neural networks is its learning capability: without programming a neural network can automatically learn to solve problems from training examples, such as

$$D_N = \{(\mathbf{x}_n, \mathbf{y}_n) \mid n = 1, 2, ..., N\}$$

where $\mathbf{x}_n$ is the *n*-th input pattern and $\mathbf{y}_n$ the associated target output pattern. The well-known error back-propagation algorithm adapts the weight vectors iteratively, that is by i) presenting an input pattern $\mathbf{x}_n$ to the input layer of the neural network, ii) computing the neural network output vector $\mathbf{f}(\mathbf{x}, \mathbf{w})$, iii) computing the error $E_n(\mathbf{w})$ between the actual output $\mathbf{f}(\mathbf{x}_n, \mathbf{w})$ and the target output $\mathbf{y}_n$

$$E_n(\mathbf{w}) = \frac{1}{2}\left\|\mathbf{y}_n - \mathbf{f}(\mathbf{x}_n, \mathbf{w})\right\|^2$$

iv) changing the weight values w_{ji} by a small amount (η) in the direction of reducing the errors, $-\frac{\partial E_n}{\partial w_{ji}}$, with respect to the weights

$$w_{ji} \leftarrow w_{ji} - \eta \frac{\partial E_n}{\partial w_{ji}}$$

and v) by propagating the errors computed sequentially and backwards from the output layers to the direction of input layers. Due to its learning capability, neural networks have been successfully used for many applications in artificial intelligence, including pattern recognition, computer vision, natural language processing, and robotics. Neural networks have also been used as computational tools for cognitive science research to investigate how humans represent, process, and learn information.

Traditionally, two-layer multilayer perceptrons have been used for most applications. However, recent research shows that deep networks, i.e. a learning model consisting of many layers of processing units, can outperform shallow networks. This is interesting since theoretically a multilayer perceptron having a single hidden layer can be proven to be a universal approximator and, thus, can learn any complex nonlinear functions. Deep networks have proven especially useful if the datasets are complex such as image and speech data. There are two different classes of deep networks. One is the deep belief networks (DBNs) consisting of multiple layers of restricted Boltzmann machines (RBMs) [1–4]. An RBM consists of two layers of neurons which are fully connected in a feedforward fashion. A DBM is traditionally trained with contrastive divergence which is based on Gibbs sampling [2].

Another class of deep networks is the convolutional neural networks (CNNs). In contrast to a deep belief network, a CNN consists of partially connected network layers. That is, a CNN consists of many stacks of a convolution layer and a sub-sampling layer (Fig. 1). The convolution layer contains a number of feature maps, where each unit has a partial receptive field of the input image. That is, in a CNN, only partial, local patches of the inputs, not all of them, are connected to the next sub-sampling layer. The sub-sampling layer consists of a number of feature maps, which are scaled local averages of the respective feature maps in the previous convolutional layer. Each unit in the sub-sampling layer has one trainable coefficient for the local average and one trainable bias. The error back-propagation algorithm is used to train the convolutional neural networks [5–9].

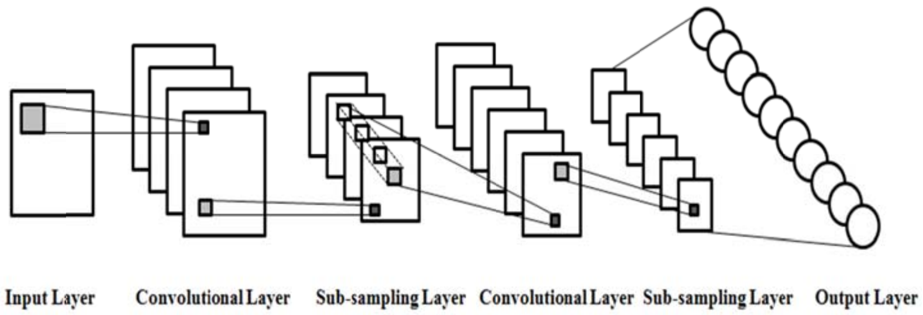

Fig.1. A six-layer convolutional neural network

3. MNIST Digit Data Sets and Experiment Design

MNIST data sets consist of handwritten digit images [10]. This database was distributed by the National Institute of Standards and Technology (NIST) [11]. The digits are centered and normalized in an image of fixed size (28×28 pixel). For the experiments, we used training set and a test set of images of single-digit numbers in ten categories (from “0” to “9”). Some example images in the database are shown in Fig.2.

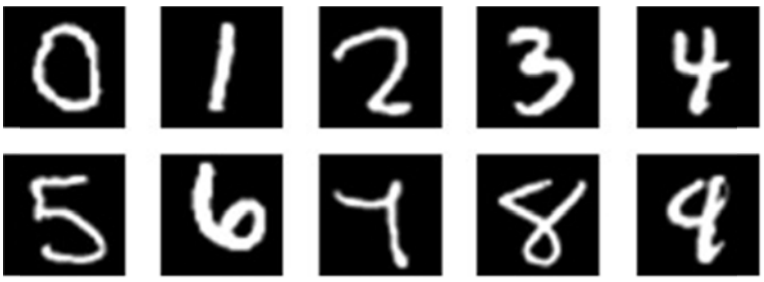

Fig. 2. Example of handwritten digit images from MNIST database.

To study the effect of model complexity on the generalization performance, we designed the experiments by varying the number of layers and units. We compared the performances (the training and test errors) of CNN models of 4 layers (Fig. 3.) and 6 layers (Fig. 1). We also varied the number of units in each layer for the 4-layer and 6-layer CNNs. We also plot the results in terms of the effective number of parameters (which depends on the number of layers and units) on the generalization performance.

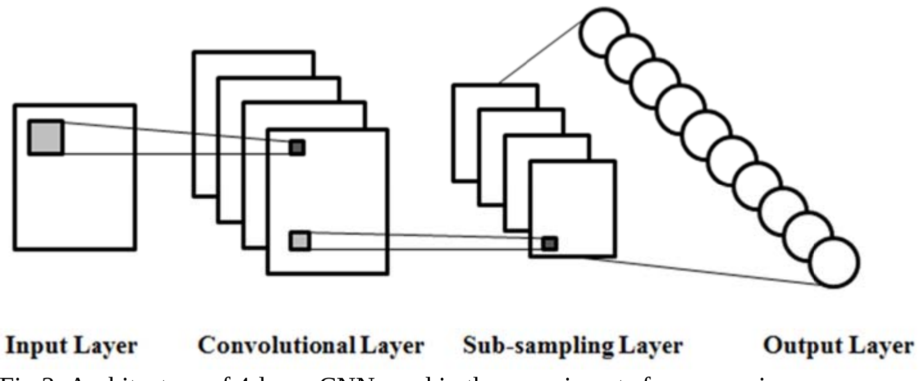

Fig 3. Architecture of 4-layer CNN used in the experiments for comparison.

4. Experimental Results and Analysis

The experiments have been performed by the Matlab code in [6]. We ran the four different kinds of experiments described in the previous section to analyze the effect of model complexity on the learning performance. The first is to analyze the training and test errors vs. the number of layers. The second is to see the training and test errors vs. the number of training iterations. The third is the change of training and test errors vs. the total number of adjustable parameters, which represents the effective model complexity. The last is the change of training and test errors vs. the total number of parameters using neural networks.

Fig. 4. shows the experimental results for the 4-layer and 6-layer CNNs. The training error is measured by the mean squared error (MSE). The test error is evaluated by the percentage of false answers i.e. the classification error rate. The training set consisted of 6,000 examples and the test set of 1,000 examples. The number of units was fixed to 8 for each layer and the number of iterations was 1,000 epochs. For this setting, we see that the deeper networks achieve better performance both in training and test errors.

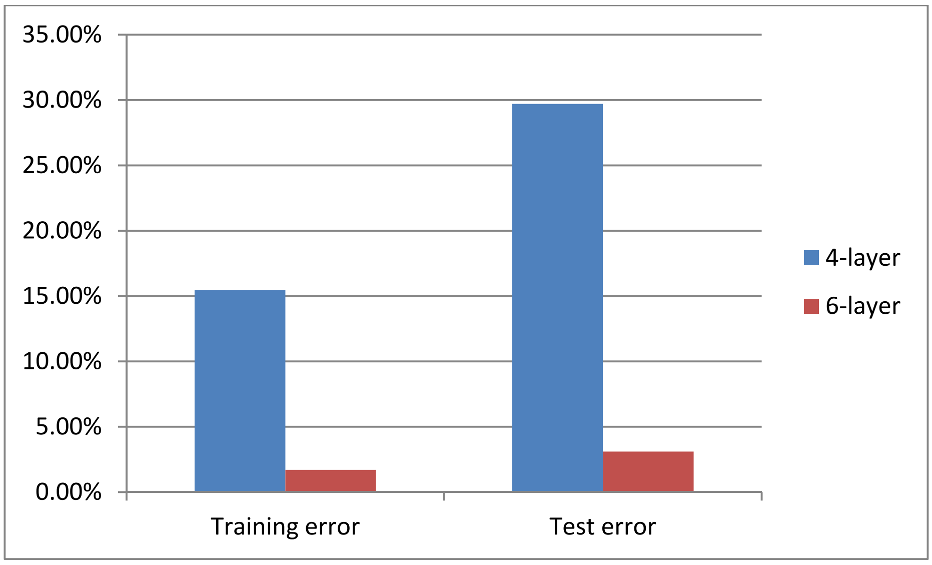

Fig. 4. The training and test errors vs. the number of layers in the convolutional neural networks.

Fig. 5 shows the changes of performance for various numbers of training epochs. Since a large number of training iterations is equivalent to a more complex model than a small number of iteration, increasing the number of iterations has the same effect of using more complex models. In Fig. 5 we see the performance gets increased as more iterations are performed. We do not observe any overfitting behavior in this setting of experiments. This shows an interesting property of the convolutional neural networks. We will come back to this issue in the discussion.

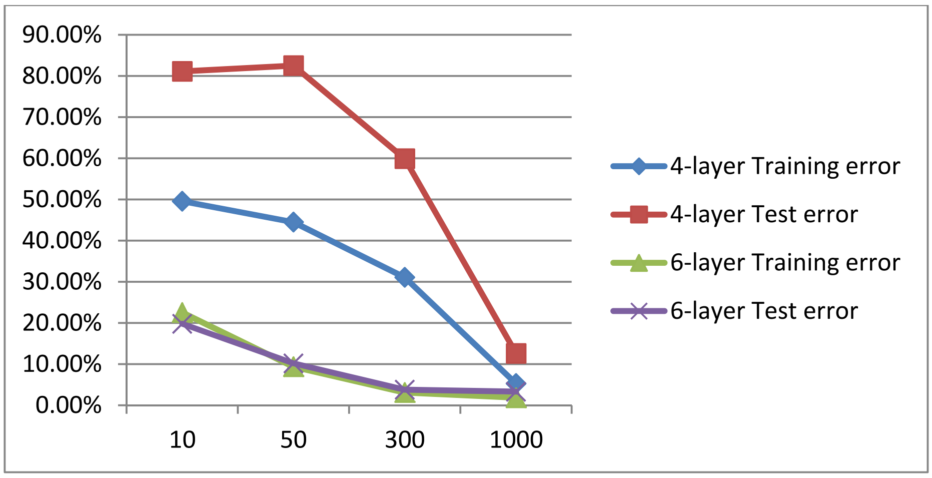

Fig. 5. Performance comparison for the number of epochs in training.

.

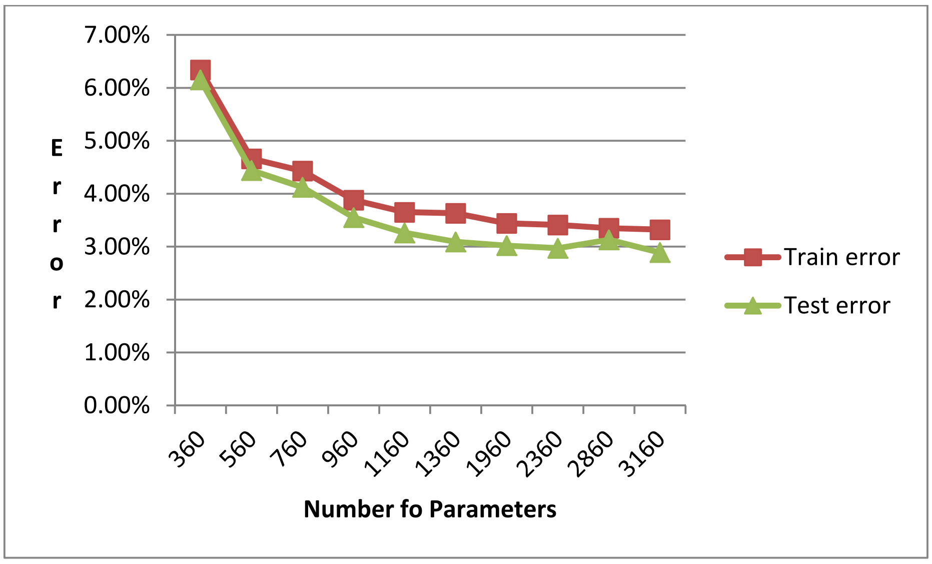

Fig. 6. Effects of the number of parameters of the convolutional neural networks (by changing the number of units as well as layers) on the training error (the mean squared error) and the test error (the classification error rate).

Fig. 6 compares the performance of the convolutional neural networks for a wide range of the number of parameters (360 to 3160). The range was generated by changing the number of units and the number of layers. The results shows that both the testing and training errors decrease as the number of parameters increases, i.e. consistent behavior of the training error (the mean squared error) and the test error (the classification error rate). That is, the generalization performance over a wide range of model complexity is good except for some specific parameter settings (e.g. around the number of parameters of 2860). This demonstrates the robustness of CNNs in generalization performance. However, this result should be interpreted more carefully. The minimum value of test error we achieved in these experiments is above 3% which is larger than reported optimal value such as 0.83% [4]. On the other hand, we observed the overfitting range experimented by neural networks(Fig. 7). In this experiment, we fixed the input nodes(784), output nodes(10) and varied hidden nodes(from 100 to 280). The results shows that the training errors continuously decrease as the number of parameters increases, but the testing errors increases around the number of parameters of 119100. This means that NNs is much weaker at overfitting problem than CNNs.

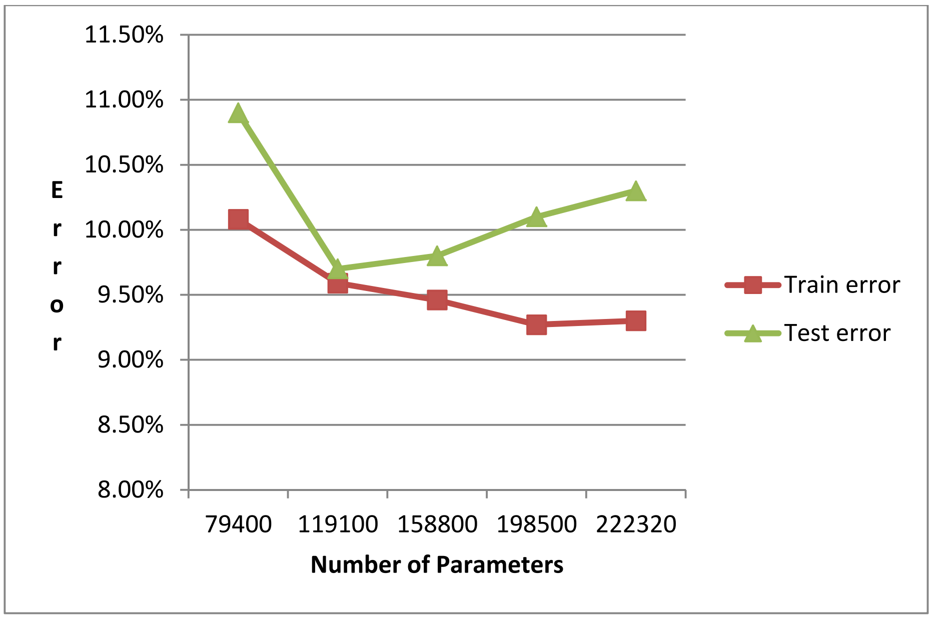

Fig. 7. Effects of the number of parameters of the neural networks (by changing the number of units as well as layers) on the training error (the mean squared error) and the test error (the classification error rate).

5. Conclusion

We studied the generalization behavior of convolutional neural networks for various combinations of model complexities. The model complexities were varied by the number of layers, the number of units, the number of epochs, and the number of effective parameters. On the MNIST digit data set of 6,000 images we found that the

CNNs show robust generalization performances over a wide range of model complexities. The results are very interesting since the machine learning theory of model complexity says that the model should overfit if the model complexity increases. Perhaps one reason for not overfitting is that our experimental settings were, despite the various settings, not wide enough to detect the overfitting ranges. And it is hard to find overfitting ranges because of sub-sampling layers. However, considering the nature of the digit recognition problem and the relatively big data set size (6,000 images for training and 1,000 images for test), the convolutional networks seem to be able to find invariance in the images. In addition, the capability of subsampling also seems to contribute to generalization performance by building sparse models of the data.

Acknowledgement

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (NRF-2010-0017734-Videome), supported in part by KEIT grant funded by the Korea government (MKE) (KEIT-10035348-mLife, KEIT-10044009).

References

1. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science, 313, 504–507 (2006)
2. Bengio, Y.: Learning deep architectures for AI. Foundations and Trends in Machine Learning, 2, 1–127 (2009)
3. Hinton, G.E., Osindero, S., Teh, Y.-W.: A fast learning algorithm for deep belief nets. Neural Computation, 18, 1527–1554 (2006)
4. Deng, L., Yu, D., Platt, J.: Scalable stacking and learning for building deep architectures. In: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2133–2136. IEEE Press, New York (2012).
5. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten Zip code recognition. Neural Computation, 1, 541–551 (1989)
6. Palm, R.B.: Prediction as a candidate for learning deep hierarchical models of data. Technical Report, Technical University of Denmark (2012)
7. Ciresan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep big multilayer perceptron for digit recognition. In: Montavon, G. et al. (eds.) Neural Networks: Tricks of the Trade, 2nd edn., LNCS 7700, pp. 581–598. Springer, Heidelberg (2012)
8. Raiko, T., Valpola, H., LeCun, Y.: Deep learning made easier by linear transformations in perceptrons. In: 15th International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 924 –932. (2012)
9. Bouchain, D.: Character recognition using convolutional neural networks. Technical report, Institute for Neural Information Processing (2006)

10. León, G.M., Moreno-Báez, A., Magallanes-Quintanar R., Valdez-Cepeda, R.D.: Assessment in subsets of MNIST handwritten digits and their effect in the recognition rate. Journal of Pattern Recognition Research, 2. 244–252 (2011)
11. LeCun, Y.: The MNIST database of handwritten digits, Available: http://yann.lecun.com/exdb/mnist/index.html
12. Wilson, K.G.: The renormalization group and critical phenomena. Review of Modern Physics, 55, 583–600 (1983)