Machine Learning Practical
Coursework 2

Jin Xu
UUN: s1673820
Exam NO: B096012

1

1. Introduction

MNIST data set is a well-known benchmark to compare different machine learning architectures, test the effects of specific techniques and investigate the possible reason behind a phenomenon. This report will focus on some of these architectures and techniques, and aims at achieving state-of-the-art on this task.

First, we will investigate different choices of non-linear transformation, including logistic sigmoid, hyperbolic tangent and rectified linear unit. Non-linear units are proved to have significant effects on deep neural networks[Maas et al. 2013]. They may affect the training performance, speed of convergence, and may behave differently in terms of saturation problem.

Convolutional neural network (CNN) has achieved great success in the field of visual tasks, due to its translation-invariance property. However, lots of efforts are put into investigating the possible modification of the original structures. Many of them have successfully improved the performance and are applied in various tasks. We will explore different settings of traditional convolutional neural network, such as the depth of architecture, the dropout technique and pre-training. We will also compare different CNN architecures in terms of their performance on the MNIST data set, and discuss possible reasons for the results.

Unsupervised pre-training is believed to acting as a good regularizer[Erhan et al. 2010], and significantly improve the performance and generalisation of a deep neural network. Auto-encoder is a very popular and easy-to-use unsupervised learning algorithm. This idea can be applied to the convolutional layer, and is called convolutional auto-encoder (CAE)[Masci et al. 2011]. We will explore the behaviour of auto-encoder on a convolutional layer and try to use it in the pre-training procedure. The learning performance will be compared to previous pure convolutional neural network.

Many boosting algorithms, which create a stronger learner from a set of relatively weaker learners, have been proposed. In this report, we will use such an algorithm called convolutional neural network committee[Ciresan et al. 2011], to combine different models together. In addition, we will test them on the MNIST data set to see if this will improve the performance on this specific task.

2. Preliminaries

We will briefly introduce the concepts and structures used in this report and reserve the detailed experiment settings to the following sections.

2.1. Non-linear transformation function

Non-linear transformation units are critical in a neural network architecture, since multiple layers of affine transformation will still end up with a affine transformation. With non-linear units, multi-layer neural network is capable of representing complex functions. However, certain function forms may behave very different in the training, in terms of the saturation problem and the shape of learning curves.

2.1.1. Sigmoid Function

Sigmoid function has the form of:

$$a = Sigmoid(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

When the input z is far away from zero, the activation value returned from this function will be very close to 0 or 1, and has very small gradients. This will pose a potential threat to the neural network training. Once our parameters are trapped into a region where z has a large magnitude, the gradient descent will have very small update, resulting in a phenomenon called saturation. Another important property of sigmoid function is that, the return value is always positive. So we will never get negative features. However, the function is differentiable everywhere and the change of gradient is continuous.

2.1.2. Hyperbolic Tangent Function

The hyperbolic tangent function is quite close to sigmoid function in terms of function shape. It has the form of:

$$a = Tanh(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}} \tag{2}$$

The hyperbolic tangent function also has the problem of saturation. The most important reason for people to use tangent function is that, this function can give us negative features.

2.1.3. Rectified Linear Unit

The rectified linear unit has very simple formula as:

$$a = max(0, z) \tag{3}$$

It has recently been proved[Maas et al. 2013] to be able to help deep learning and do not suffer from the saturation problem. However, the gradient will have a rapid change when z jumps between the positive and negative region, even the change of inputs is quite small.

2.2. Convolutional Layer

Convoluional layer introduces weights-sharing and local receptive fields into the full-connected layer. If you know the normal neural network, the forward propagation of a convolutional layer has very similar form, with matrix multiplication replaced by convolution.

If we have inputs matrix x, and kernel weights W, the forward activation can be written as:

$$h = x * W + b \tag{4}$$

(the $*$ denotes convolution) Where a kernel defines a certain feature maps, and we can have multiple feature maps corresponding to different features extracted from a local area. The biases term is shared across the whole feature map.

2.3. Max-pooling

It is common to insert a max-pooling layer after a convolutional layer. The max-pooling layer just pick the largest value over a certain area, for example a 2×2 small filter. The use of max-pooling will significantly reduce the number of hidden units, thus prevent model from over-fitting. To be notice, max-pooling can be seen as a non-linear transform just as rectified linear unit.

2.4. Auto-encoder(AE)

An auto-encoder stacks an encoder which maps the inputs into a feature space, and a decoder which tries to recover the original inputs from features. The normal architecture is a neural network with one hidden layer, trying to represent an identify function. Since there are non-linear transformations in this architecture, the hidden feature layer can learn some interesting features from inputs.

There is a variation of auto-encoder called denoising auto-encoder(DAE). Certain noises are added to the inputs, so that the feature layer is forced to learn meaningful features. A common noise schema is randomly setting part of the inputs to 0, which will be adopted in this report.

2.5. Convolutional auto-encoder

The unsupervised feature extraction of an auto-encoder and the transition-invariance character of a convolutional neural network can be combined together, and form a so-called convolutional auto-encoder(CAE). It is just an auto-encoder with both the encoder and decoder being convolutional layers. However, different feature maps are combined together to recover the original inputs. The k-th feature map can be written as

$$h^k = \sigma(x * W^k + b^k) \tag{5}$$

The W^k and b^k are the kernel and biase term of a feature map, and the $*$ denotes convolution again. Certainly, the output layer will have only one feature map, and has the form of

$$y = \sigma(\Sigma_k h^k * \bar{W}^k + c) \tag{6}$$

where the $\bar{W}^k$ is kernels of the decoder. So the error function is just the mean-square-error of x and y.

2.6. Convolutional Neural Network Committees

Convolutional neural network committee for MNIST task is motivated by a simple notion that different writing styles has different aspect ratios. As stated in [Ciresan et al. 2011], we can normalise image width to different pixels and different convolutional neural networks are responsible for specific writing styles. The final prediction is just an average of all the models.

3. Learning Architecutre

3.1. Data Set

For our experiments, **MNIST** handwritten digits data set is used for training and evaluation. The data set contains labeled 18×18 images which can be classified as 10 digits. The whole data set is shuffled and divided into training set and validation set. We use 50000 images for training, and the remaining 10000 images for validation. During the training procedure, mini-batch stochastic gradient descent is used and the batch size is fixed to 100.

3.2. Models

All the experiments in this report will be conducted using certain kinds of convolutional neural network, and a 2×2 max-pooling layer is inserted after every convolutional layer. The kernel filters in each convolutional layer are 5×5, and use a stride of 1. A final full-connected layer and a softmax layer will be stacked onto the convolutional structure to get the final prediction.

Most of our experiments will use a shallow convolutional neural network, with only one convolutional layer. So there will be 4 feature maps with a size of 24×24. A deeper version will have two convolutional layers, and the second layer has 12 feature maps of a 4×4 size.

However, the structure of convolutional auto-encoder is quite different. To make sure the output has the same size of input, full padding will be applied to hidden feature layer. So the input and output layer will have the same shape of 28×28, while the feature maps are 24×24. We will still use 4 feature maps in this auto-encoder, and all the feature maps will be combined together to get final outputs.

The convolutional neural network committee consists of 3 convolutional neural network with one convolutional layer. The width of 28×28 images are normalized to 20, 24, and the original separately. The final prediction is the average softmax outputs of all the models.

3.3. Initialisation, Learning rule and Training

To keep the scale of activations at different layers of the network the same at initialisation, a newly proposed initialisation scheme[Glorot and Bengio 2010] is applied to the weights of full-connected layer at the beginning. However, biases are all initialised to constant 0. All the convolutional kernel weights are uniformly initialised between -0.01 and 0.01.

All the training procedures in this report go through a total of 30 epoch, and each epoch traverses the whole training set using a batch size of 100. We use back-propagation to calculate our gradients efficiently.

To accelerate the speed of convergence, we use the momentum learning rule, with a 0.9 initial learning rate, and a 0.02 momentum coefficient.

3.4. Evaluation

Since we are dealing with one-hot encoding targets classification problem, we will use cross-entropy error function to measure the difference between outputs and targets. The error will be averaged on the whole data set so that we can easily compare performance between training set and validation set.

The error function measures how likely the data set is being observed given the model and parameters. It is a good measure includes our confidence of the model prediction. However, another measure called accuracy can be used to evaluate and compare model performance. It can be easily calculated as:

$$acc = \frac{R}{S} \tag{7}$$

where R the number of correct predictions and S is the evaluation set size.

4. Experiments

4.1. Non-linear Transformations

As a comparison, we train the same neural network architecture(shallow convolutional neural network, as stated in 3.2) and replace only the non-linear transformation units. As you can see in figure 2 and figure 3, the sigmoid function seems to slow down the training procedure, and still does not converge in the visible time window. On the other hand, the hyperbolic tangent and the rectifier linear unit(ReLU) perform well, with ReLU slightly better. The 'None' stands for no non-linear units. Since there is max-pooling layers in our architecture, and it can be seen as a kind of max-out non-linear unit, so it is possible to remove all the other non-linear units away. A possible explaination for the slowness of sigmoid function is that sigmoid is much smoother than hyperbolic tangent, see figure 1, even though they have very similar properties. When the input is in the near-zero region, the gradient of hyperbolic tangent and ReLU is much larger than sigmoid function. So in order to make sigmoid function working properly, a larger update step should be used.

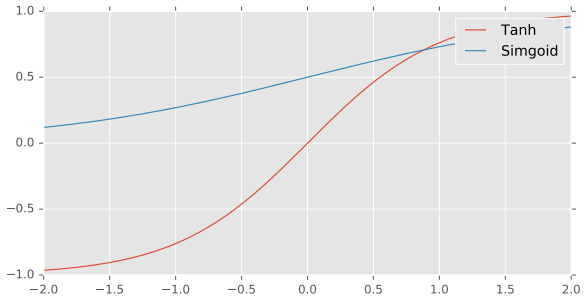


Figura 1. sigmoid and tanh function

It is worthwhile to mention that the learning curve of sigmoid function and hyperbolic tangent are very smooth. While there may be an abrupt increasing or decreasing in case

of ReLU and pure max-pooling. It is explainable because the the gradients of sigmoid and hyperbolic tangent are also differentiable, so their updates using gradient descent will not have a sudden change. In terms of ReLU or pure max-pooling, the gradient may jump between 0 and 1, even with only a small change of inputs. This can be seen as a downside of ReLU. That is to say, this unit is not stable.

We cannot observe significant saturation during the training procedure. Our weights initialisation schema seems to work very well. The average activations of the first convolutional layer and full-connected layer with error bars are provided in figure 4 and figure 5. The average activation is very close to 0.5 in terms of sigmoid function and very close to 0 in terms of hyperbolic tangent. Both of their spread is acceptable. As you can see, the activation of ReLU can go very large, since their values are unbounded.

In the following experiments, we will set all of our non-linear units to be hyperbolic tangent function, due to its speed of convergence and stable gradient updates.

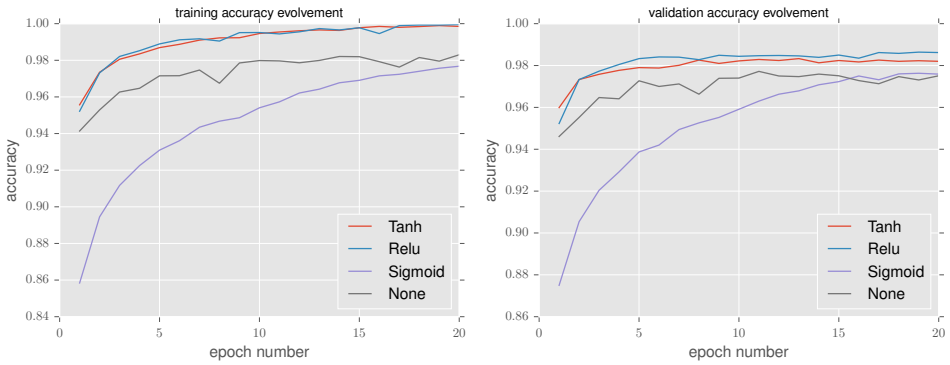


Figura 2. non-linear units accuracy evolvement

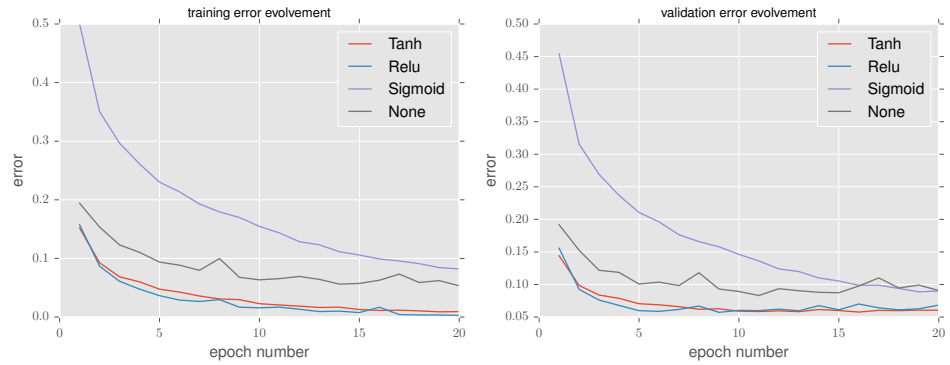


Figura 3. non-linear units error evolvement

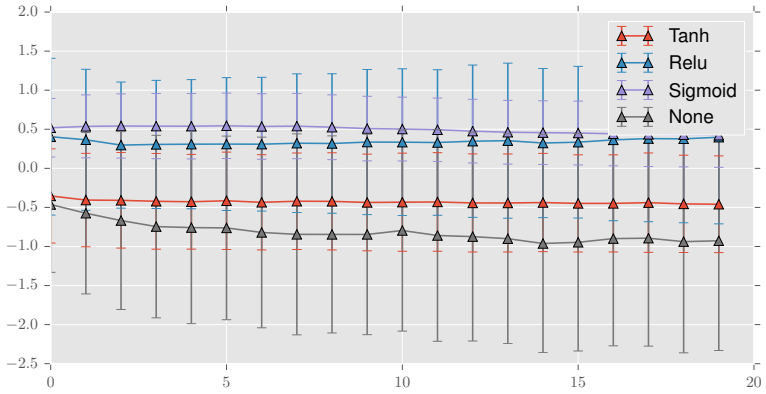


Figura 4. convolutional layer activation

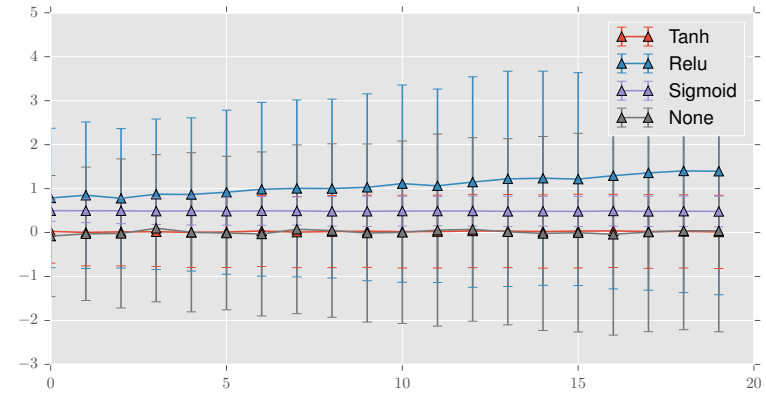


Figura 5. full-connected layer activation

4.2. Convolutional neural network

4.2.1. Shallow convolutional neural network

As a beginning, a direct experiment is conducted on the shallow convolutional neural network, and is compared to the performance of full-connected multi-layer neural networks. As shown in figure 6, the convolutional neural network achieves an accuracy of 98.6% on the validation set, which is a significant improvement of a full-connected structure(97.9% in the last report). The training is basically finished after 20 epochs, but the training time of each epoch is significantly longer than a full-connected one.

In conclusion, the convolutional layer does improve the overall performance on the MNIST vision tasks. And we will explore different techniques, which can be applied to this simple shallow CNN in the following parts.

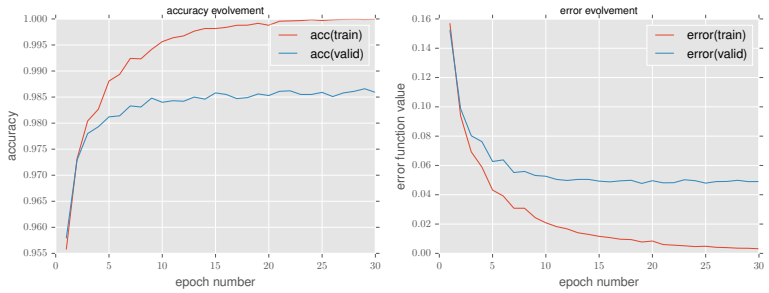


Figura 6. shallow CNN training evolvement

4.2.2. Dropout in convolutional neural network

Dropout is a popular technique used in the neural network, to prevent model from over-fitting. As shown in figure 6, the training error goes to 0 in the end, which is a sign of over-fitting. So we conduct a new experiemnt using dropout in the shallow convolutional layer. A 50% dropout is applied to the inputs and to the final full-connected layer, while a 10% dropout is applied to the convolutional layer. The experiment result is shown in figure 7. As shown, the zigzag of the error and accuracy evolvement curves indicate that the training procedure is quite noisy, and ending up with a pool final prediction performance .

The weights-sharing is the main property of convolutional layer by which it is distinguished from normal neural network layer. So there can be an argument that the convolutional layer needs less regularization. A common regularizer to normal layers may be too strong to a convolutional layer. However, we already set the dropout rate to 0.1, so the influence should be small. An interesting phenomenon is that the training and validation error is quite close. That is to say, the dropout model will generalize very well.

One possible future investigation is that, we should set a dropout rate scheduling. At the beginning, the dropout rate is large, so that our model will have good generalzation. While in the end, the dropout rate is small, so a quicker convergence can be possibly achieved.

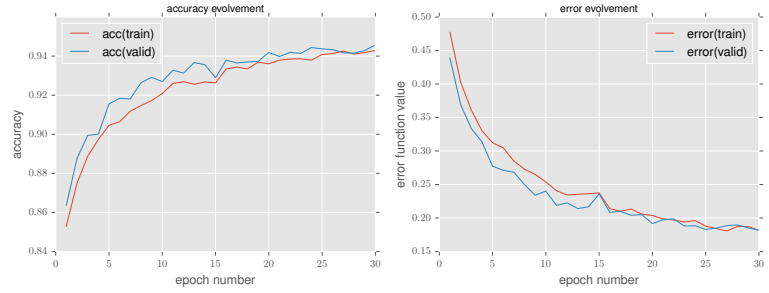


Figura 7. shallow CNN dropout training evolvement

4.2.3. Going deeper

The depth of a neural network has important influence on the learning procedure, So I want to see if a deeper convolutional neural network will have better performance. The architecture is stated in 3.2, and the evaluation result is shown in figure 8

The final accuracy on the validation set is 98.7%, which is only slightly better than a shallow one(98.6%, full comparison in table 2). On the other hand, the training time is 4 times of the shallow one.

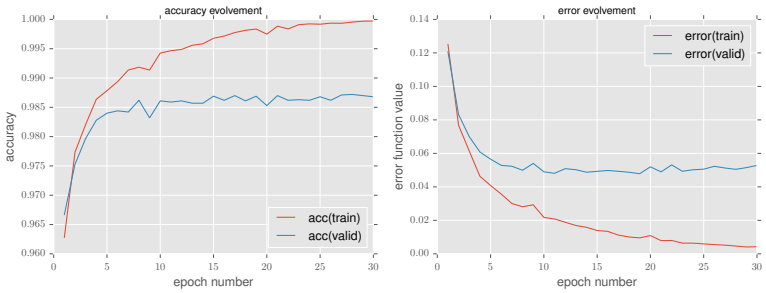


Figura 8. deeper CNN training evolvement

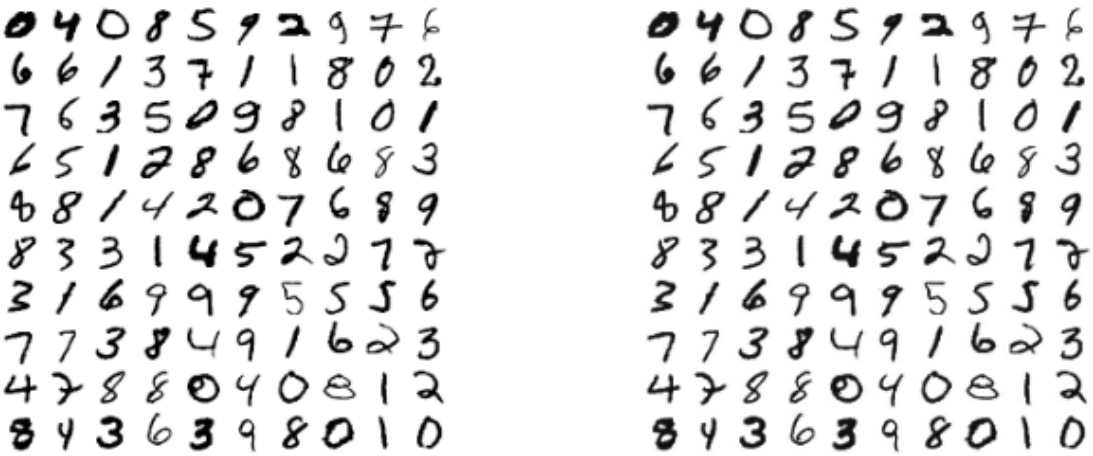
4.3. Convolutional auto-encoder

The local connection structure of convolutional layer makes it very suitable for image feature representation. And we know that auto-encoder can learn interesting features from input images without labels. So will convolutional auto-encoder achieves a better automatic feature extraction? We train two CAE models, and the second one is the denoising version, in which inputs are randomly corrupted (set to zero). The inputs corruption rate is set to 0.5. Unlike other models in this report, the momentum initial learning rate is set to 0.0002, which is far smaller than our default setting. That is because during the training of CAE, significant saturation is observed, and all the activations are very close to asymptotic values. The activations seem to explode to during the forward propagation. After proper hyper-parameter tuning, our models get very impressive outcomes.

As you can see in figure 9 and figure 10, the convolutional auto-encoder almost fully recover the original inputs from its features, and the 4 feature maps have relatively diversified representation. The inputs of the noisy version is heavily corrupted, but the convolutional denoising auto-encoder still has very good performance(figure 11, 12), making us believe that the feature layer has learn some interesting features instead of remembering the data.

4.4. CAE as pre-training

It is known that unsupervised pre-training helps the performance of neural network. So will convolutional auto-encoder improve the performance of our convolutional neural network on the MNIST data set? We use the trained parameters from CAE(Convolutional Auto-encoder) and CDAE(Convolutional Denoising Auto-encoder) as an intialization of the first convolutional layer parameters, and get the results in figure 13 and figure 14. Unfortunately, the results is a little worse than pure convolutional neural network, and the factor of noise has no effect on the outcomes.

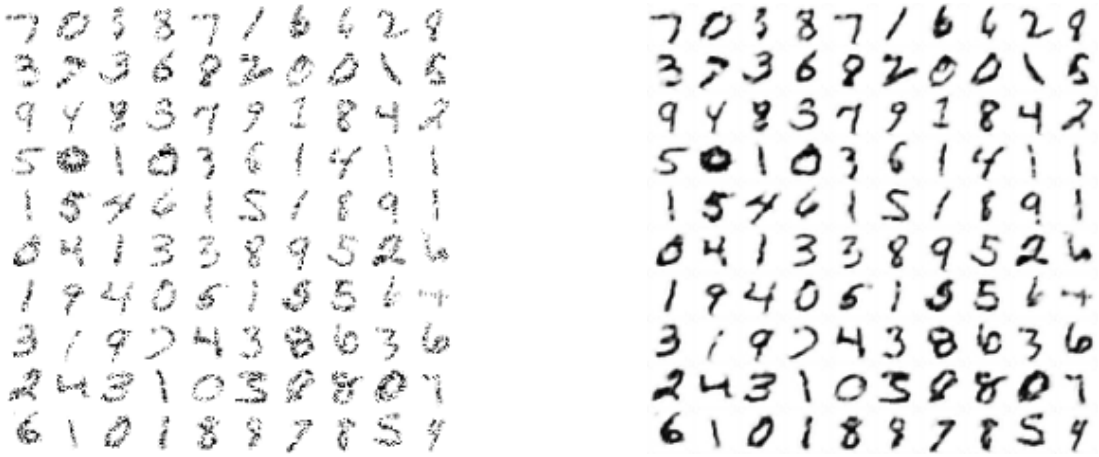


(a) inputs(no noise) (b) recovered images

Figura 9. inputs and recovered images

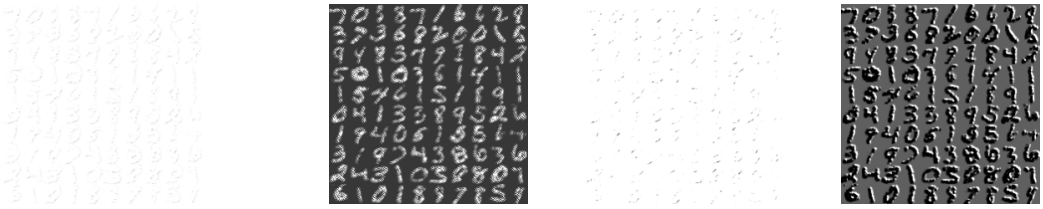
(a) (b) (c) (d)

Figura 10. feature maps



(a) inputs(0.5 noise) (b) recovered images

Figura 11. inputs and recovered images(denoising)



(a) (b) (c) (d)

Figura 12. feature maps(denoising)

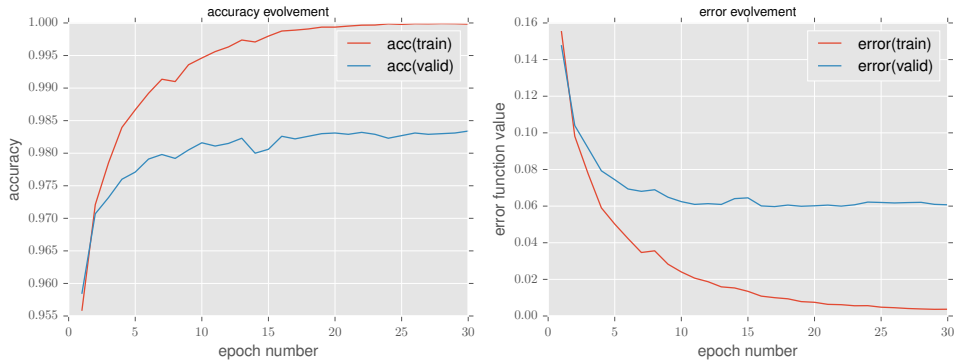


Figura 13. CAE Pre-training CNN

4.5. Convolutional neural network committee

At the end, we will investigate a boosting algorithm called convolutional neural network committee. As stated in 3.2, we will train three different shallow CNN on training set with different writing styles, and form a committee using this three models. The prediction accuracy of separate models and the combined committee model is compared in table 1.

From the table, we can draw the conclusion that CNN committee is a stronger learner than any of this model. However, the improvement is not significant.

For future investigation, a committee with more diversified models should be tested. And also, this boosting algorithm just averages over all the weaker learners. Intuitively, different models are good at different inputs, so a weighted version such as AdamBoost may potentially do a better job.

Tabela 1. CNN Committee

model	**W20**	**W24**	**W28**	**committee**
acc	97.5%	97.9%	98.6%	98.7%

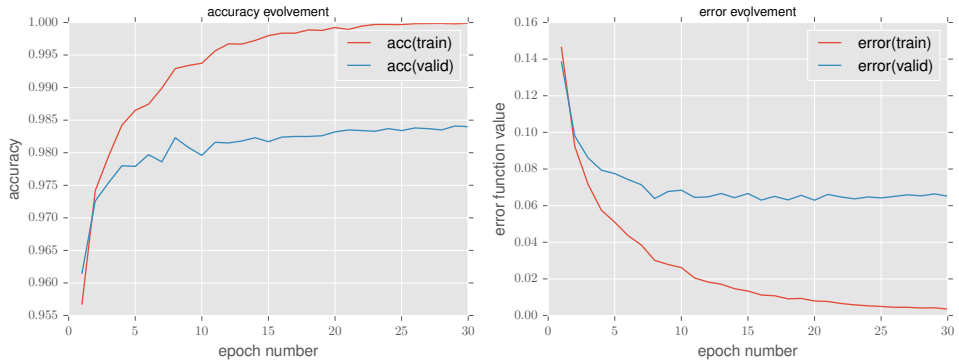


Figura 14. CDAE Pre-training CNN

Tabela 2. Model Comparison

model	**validation acc**
Multi-layer neural network	97.9%
CNN	98.6%
CNN with dropout	94.6%
CNN with DAE pre-training	98.4%
CNN with AE pre-training	98.3%
Deeper CNN	98.7%
CNN Committee	98.7%

5. Conclusion

Convolutional neural network shows great performance on vision tasks like MNIST data set. The convolutional layer structure can also be integrated into an auto-encoder, which is capable of learning interesting features directly from inputs. However, techniques like dropout and unsupervised pre-training are applied to the CNN, and no improvement is observed. Different kinds of non-linear units are compared and they all have their own advantages and drawbacks. The CNN committee is proved to improve the performance of separate shallow CNN models, however, the improvement is limited.

Referências

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. (2011). Convolutional neural network committees for handwritten character classification. In *2011 International Conference on Document Analysis and Recognition*, pages 1135–1139. IEEE.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Aistats*, volume 9, pages 249–256.

Maas, A. L., Hannun, A. Y., and Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, pages 52–59. Springer.