# Proceedings in Artificial Neural network of Autoencoder combined with Convolutional Neural Network

Zeruo LIU

u6277483@anu.edu.au
Research School of Computer Science
Australian National University

**Abstract.** Both autoencoder and the Convolutional Neural Network could get the symbolized data from the dataset. In my work, I want to boost the efficiency of feed-forward networks, and I get the Semeion dataset trained by normal sequential feed-forward, autoencoder, Convolutional Neural Network and putting the encoded data to the Convolutional Neural Network model. I compare the results of the four conditions and get the characteristics of the autoencoder and Convolutional Neural Network directly. The functionality of the compression by autoencoder will reduce the data quality, also it will prune the redundant data to avoid the over-fitting and improve the efficiency of training the dataset. In addition, the autoencoder will realize the unsupervised learning and reduce the dimensions for encoding for the dataset. At the same time, the Convolutional Neural Network improve the accuracy of the training neural network. The challenge method is the combination of them could improve and optimize the accuracy and efficiency both.

**Keywords:** Compression, Neural network, Autoencoder, Encode, Decode, Feedforward, Backpropagation Convolutional Neural Network, Weights, Pooling, Loss

## 1 Introduction

In this paper, I suppose to train the dataset named "Semeion Handwritten Digit Data Set" which is a classification problem. This dataset contains one data file with 1593 records (rows) and 256 attributes (columns) in total. I choose this dataset because the data and attributes quantity is enough for the training and also reasonable to run in a controllable time about 1 minute. The attributes number(256) is large enough to train the proper data.

The purpose of this dataset is for the commitment of classifying the handwritten digit (from 0 to 9 means the feature) of the number accurately and quickly and to test their difference respectively. This would help the accurate recognition of the handwritten and set benchmark for the handwritten recognition algorithm. Some of numbers are written quickly and some are normal writing. The commitment was to write the digit the first time in the normal way (trying to write each digit accurately) and the second time in a fast way (not accuracy) and distinguish them. 1593 handwritten digits (rows) from around 80 persons were scanned, stretched in a rectangular 16x16 in a grey scale of 256 values. Then each pixel of each image was scaled into a boolean value as the feature using a fixed threshold [1].

The early research method on this dataset is to test several Artificial Neural Network methods inspired by biological network and all the experiments are tested in the METANET system.

The previous handwritten research show that the representation depth is beneficial for the classification accuracy, and that state-of-the-art performance on the ImageNet dataset can be achieved using a conventional ConvNet architecture (LeCun et al., 1989; Krizhevsky et al., 2012) with substantially increased depth.[2] Convolutional neural network is commonly used in the visual image, so I take this feedforward deep learning method.

In my study, the feedforward neural network with multiple layers, 30, 256, 60 neurons for each hidden layer is trained for classification of the numbers . Then I implement the auto-encoder according the technique of the auto-associative topology in *Image Compression using Shared Weights and Bidirectional Networks* [3] research for dimension reduction and data pruning. After that, I get the smaller sized pruned data trained in the two-layer Convolutional Neural Network (CNN) model to improve the detection performance significantly. The combination of also show the efficiency of handwritten recognition both in speed and accuracy.

## 2 Method

### 2.1 Feed-forward neural network

I use the the feed-forward network with backpropagation, momentum and softmax to train. The result is stable and the testing accuracy is around 90%, which is quite close to the final result of the relevant paper on the UCI repository .

The neural network I implement is reading from the raw dataset and set a sequential network with 3 layers(30, 256, 60 hidden neurons for each). To improve the accuracy of the training, the LeakyRelu plays better as the other advanced activation layer. Following format is the LeakyRelu and difference from the Relu is when x ≤ 0, returns 0.

$$\frac{\partial E}{\partial x} = \begin{cases} \nu \frac{\partial E}{\partial y} & \text{if } x \leq 0 \\ \frac{\partial E}{\partial y} & \text{if } x > 0 \end{cases}$$

As it can be seen in the figure 1, this topology is one directional forward from input to output and does not form a cycle a loop in the neural network. I connect these layers together with the mentioned Leaky Relu as activation function subsequently into a multi-perceptions structure.

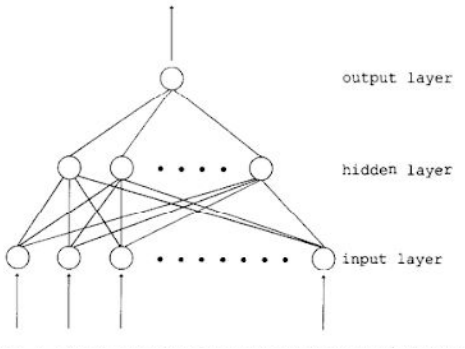


**Fig. 1.** the Topology of the feedforward neural network[4]

### 2.1.1 Back-propagation

The back-propagation algorithm is used in feed-forward network. The backpropagation is used to the derivatives of the function for the output layer and then for the hidden layers can be recurrently analyzed as the following subtraction. [5]

$$\omega_{ij}^{(k+1)} = \omega_{ij}^{(k)} - \lambda \left( \frac{\partial E}{\partial \omega_{ij}} \right)^{(k)}$$

$$\vartheta_i^{(k+1)} = \vartheta_i^{(k)} - \lambda \left( \frac{\partial E}{\partial \vartheta_i} \right)^{(k)}$$

wij means the coefficient weight of the ith neuron and the jth neuron. vi means the coefficient threshold of the ith neuron. λ means the learning rate. The division after that means the derivatives.

Each propagation involves:[6] Generating for the outputs, calculating the error of the outputs and back propagation of the output to calculate the delta (delta = the targeted value - current value) .

In my implementation, the Adadelta optimizer algorithm and the weight decay are used for each iteration.

### 2.1.2 Loss function

I choose the cross-entropy to calculate the loss than display them in graph and table.
The Cross Entropy method involves two steps for each iteration:[7]

1. Calculate new data sample randomly according to a specified mechanism.
2. Update the parameters of the random mechanism based on the previous data to produce a better sample for next iteration then repeated

### 2.1.3 Splitting dataset

The normal split for train, test and validation is 70%, 15% and 15%. As the optimal dataset description tested and shown, I also choose the 50% of the test dataset and 50% of the train dataset. There is no sign to show the over-fit. The **validation set** to split some data will lose the training data and the test data and the data is not abundant enough. Considering the effect of the validation is not good, I control the number of the epoch to eliminate the over-fit instead. The split of the test and training data is following the dataset repository page giving the suggestion to split it half to half randomly. I test it the half half split reaches highest of all over 90% testing accuracy.

## 2.2 Auto-encoder

Compared to the paper named “Image Compression using Shared Weights and Bidirectional Networks”, I pick out the method of **auto-encode** in an auto-associative topology, because I want to reduce the pressure of the information the neural network receive. The figure 2 is the topology of encoder(left side) and decoder(right side).

First is to encode and decode the data from left to right. After that, comparing the original data and depressed data to give the prediction loss. The loss is processed by back-propagation to improve the accuracy of autoencoder. The key is the middle of the hidden layer containing the symbols of the data. Obviously, I only use the original data to realize the unsupervised learning.

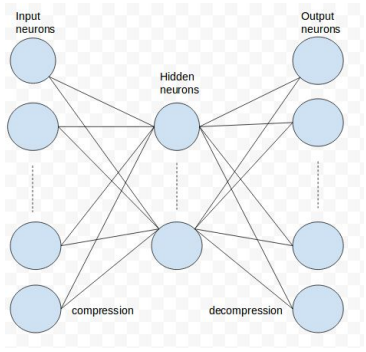


**Fig. 2.** the Topology of the autoencoder

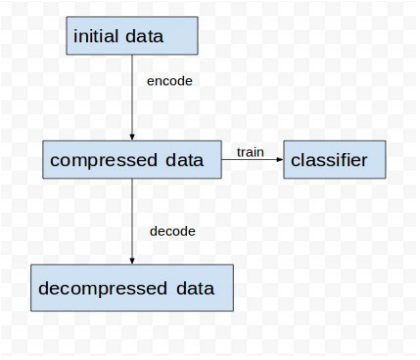


**Fig. 3.** Compressed process

### 2.2.1 Autoencoder Definition

I use the undercomplete autoencoder to message the data to realize the unsupervised training and compress the data during encoding while decompress during decoding as figure 3 process.

The encoder is to transform the high-dimensional data (256 data attributes) into a low-dimensional (16 data) code and decoder network is to recover the original data in the same dimension(256 data attributes) from the encoded one[8]

Auto-associative network topology is to keep the input and output neurons in the same number while the hidden neurons are less than that number. In this case, the input and output keep 256 as the figure 4 shown.

The significant feature of training an autoencoder network for handwritten digit compression is that the hidden neurons train a compressed representation for the input models. For passing over an inefficient network connection, the neural network weights could be transmitted as a fixed initial cost, while subsequently the compressed representation activated by the hidden neurons which could be passed over by using the activation values of the hidden units for the handwritten image. All neural compression is persistently lossy.[9]

The image with 256 pixels is preprocessed to 64, 32 then 16 neurons gradually. Then the encoded data are message to the classifier to test the loss of the compression. The classifier is set to true to evaluate the encoded information and then write into a new data file for comparison. Vice versa, setting false will continue to decode the data and training for the encoded data. The way I define encoder and decoder is to use the sigmoid activation function, so the compressed value should be from 0 to 1. The activations I use in the decoder are corresponding to that in the encoder.

The experiment is to compare results to the original training without auto-encoding. The comparison of the encoded is to show the loss of compression. The encoded data is compared to demonstrate the training and testing effect of auto-encoded data.

### 2.2.2 Merits and Demerits

It does help in decreasing the dimension and boost efficiency of the training after compression, because the compression prunes the redundant data and pick out the most dramatic information to train.
However, if ignoring too many data to excess compression, the accuracy decreases and losses increase. At the same time, if pruning most of the redundant and futile data, the accuracy will increase on the contrary.

The theoretical advantage of this procedure is compared due to the fact that the successes, failed blows, false attributions, and correct eliminations. In addition, the obsessions, removals, predictions and simplifications of the Artificial neural network are computed in a parallel system instead of through the calculation of variables groups. On the contrary, the demerits is that this autoencoder method takes a great amount of computation time.[10]

## 2.3 Convolutional Neural Network(CNN)

```
nn.Conv2d(1, 16, kernel_size=5, stride=1, padding=2),
nn.BatchNorm2d(16),
nn.ReLU(),
nn.MaxPool2d(kernel_size=2, stride=2))
```

**Code. 1.** the layer structure of CNN

CNN is consisted four parts in each layer, convolutional layer, rectified linear units layer, pooling layer and loss layer in total for the hidden layer. The CNN structure in pytorch is shown as code 1 and the construction details is demonstrated in Figure 4. As same as the normal feedforward network, it contains an input, one or several hidden layers and an output. CNN is more accurate because the network will learn as a subpart of the image feature instead of the less symbolized features pixel by pixel. After rolling around the whole handwritten picture by batch filter , the border information is acknowledged also enhance the deep learning.
Additionally, the backpropagation and the loss functions are same in my CNN structure training.

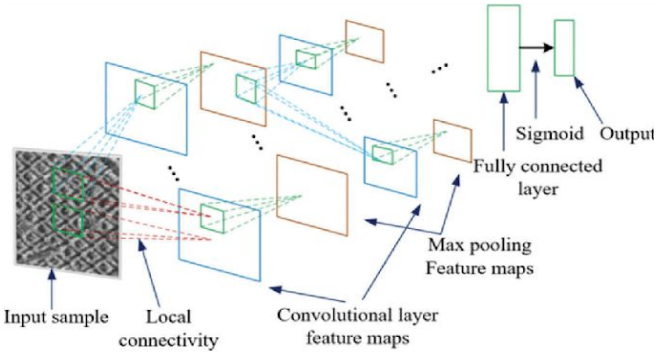


**Fig. 4** the Topology of CNN[11]

### 2.3.1 Convolutional Layer

There are many convolutional units in a convolutional layer. The convolutional units parameters are optimized by the backpropagation introduced in 2.2.1. This is the first layer in the CNN to extract the simple characteristics to the complicated one from iteration to iteration.

### 2.3.2 Rectified Linear Units layer (ReLU layer)

Choosing this method will keep the sparse and decentralized neural network be activated. Also, RELU correction and regularization can be used to debug the activity of the neurons in the machine neural network to output positive value. It is efficient for gradient descent and backpropagation, because it avoids gradient explosion and gradient disappearance. The last point is that it simplifies the calculation process, while the dispersion of activity makes the overall computational cost of the neural network decrease.

The Relu layer use the activation of the following function:

$$f(x) = \max(0, x)$$

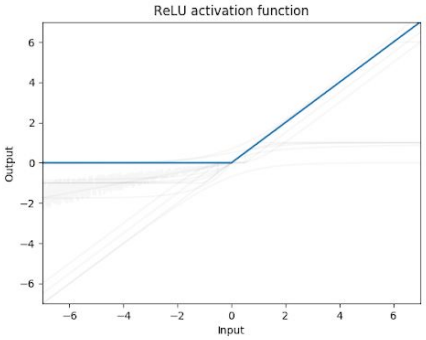


**Fig. 5** the function graph of the RELU

This layer could help to detect the border of the neural network and decrease the training time.

### 2.3.3 Pooling Layer

I use the Max Pooling to separate the sample image into several small areas. In each small area, this function return the max value from this sub-area. The accurate speciality of the image is compared with the other related characteristics. Pooling layer is to filter and extract the symbolized data to keep the accuracy and then throw the processed data in to the next layer for analyzation. Also, this pooling layer will avoid the too many parameters calculation which can decrease the quantity of the parameters calculation, save the memory and avoid overfitting to some extent.

### 2.3.4 Loss layer

The loss layer stays in the last layer of the CNN which will punish the error between the target and calculated result. This layer function is to choose the least value of the difference between target and actual value to minimize the loss. The loss is calculated by the forward direction while the loss parameters gradient is calculated by backpropagation.

## 3 Results and Discussion

To make things clear, I compare the results in four conditions combining the normal, autoencoder and CNN also putting the encoded data to CNN model.
The evaluation I use is graph(loss and accuracy), table, and matrix. The graphs could show the tendency of the data, while the tables could show the accurate of the details. My confusion matrix is used in the supervised learning to show the algorithm performance visually. Each row means the class and each column means predicted value in the matrix. The confusion matrix could test if the model is confused by several classes after classification. This matrix is quite clear without much confusion.

### 3.1 Normal feedforward neural network

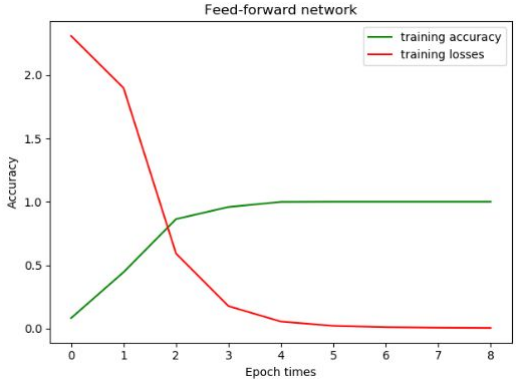


**Fig. 6.** The training accuracy and losses of normal feed-forward network

The training accuracy will increase and loss will decrease stably in the feed-forward with the epoch iterations.

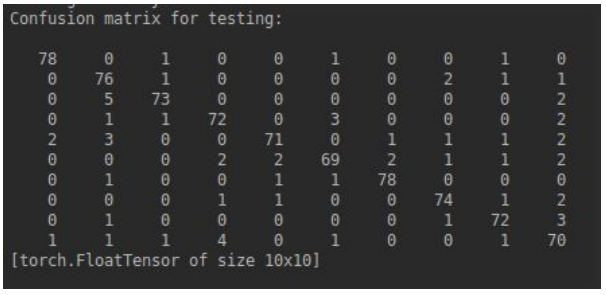


**Fig. 7.** Confusion matrix of normal feed-forward network

The number is clear for each row in Figure 7 matrix can be seen.

This dataset can be trained in simple feed-forward neural network for clear classification.

| Test time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Testing Accuracy | 91.22% | 92.34% | 91.72% | 90.84% | 90.34% | 91.22% | 91.09% | 91.47% | 91.09% | 91.25% |

**Table 1.** 9 times testing accuracy of normal feed-forward network

The testing accuracy are listed and tested 9 times and the average testing accuracy is calculated. The testing accuracy is stable around the average of 91.25%, which performs well.
Therefore, the over-fitting problem is not serious with the control of the epoch number.

| Epoch(270 in total) | Losses | Training Accuracy |
|---|---|---|
| 1/270 | 2.3053 | 10.93% |
| 31/270 | 0.2606 | 60.05% |
| 61/270 | 0.0107 | 83.54% |
| 91/270 | 0.0031 | 94.85% |
| 121/270 | 0.0014 | 99.37% |
| 151/270 | 0.0008 | 100% |
| 181/270 | 0.0006 | 100% |
| 211/270 | 0.0004 | 100% |
| 241/270 | 0.0003 | 100% |

**Table 2.** The training accuracy feed-forward network

I pick is the accuracy and loss of each 30 epoch in a 270 epoch totally in a table. The training accuracy is stable and finally reaches the 100% after training 200 epoch. It is accurate to list the data of accuracy and the losses for each test.

### 3.2 Auto-associative neural network

| Epoch | 1/600 | 51/600 | 101/600 | 151/600 | 201/600 | 251/600 | 301/600 | 351/600 | 401/600 | 451/600 | 501/600 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Loss | 0.209% | 0.098% | 0.069% | 0.067 % | 0.0543% | 0.047% | 0.045% | 0.037% | 0.042% | 0.035% | 0.034% |

**Table 3.** The loss of the compression data in the 600 epoch.

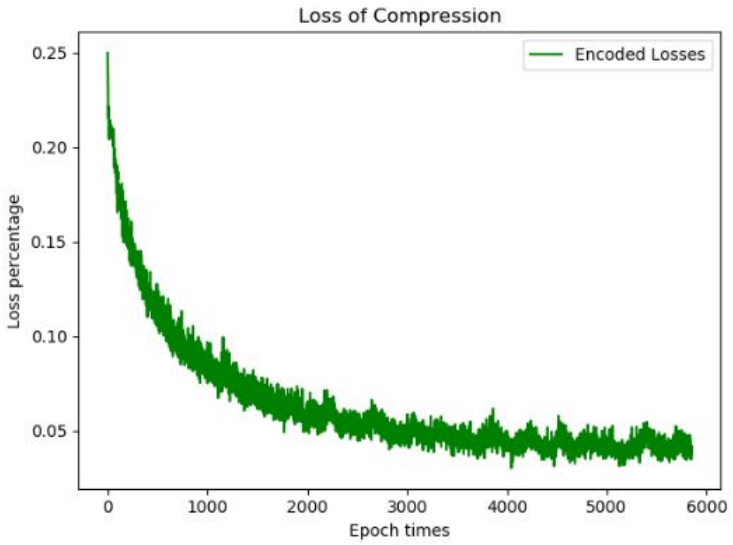


**Fig. 8.** The loss during the encoded process

The loss will also decrease in the feed-forward with the epoch iterations as mentioned before.

The loss is more fluctuated so the autoencoder will prune some key information and lose data could be concluded.

Fewer outputs for the encoder, the more extent the data will be compressed and the more data will be lost. Also this loss is irreversible and random.

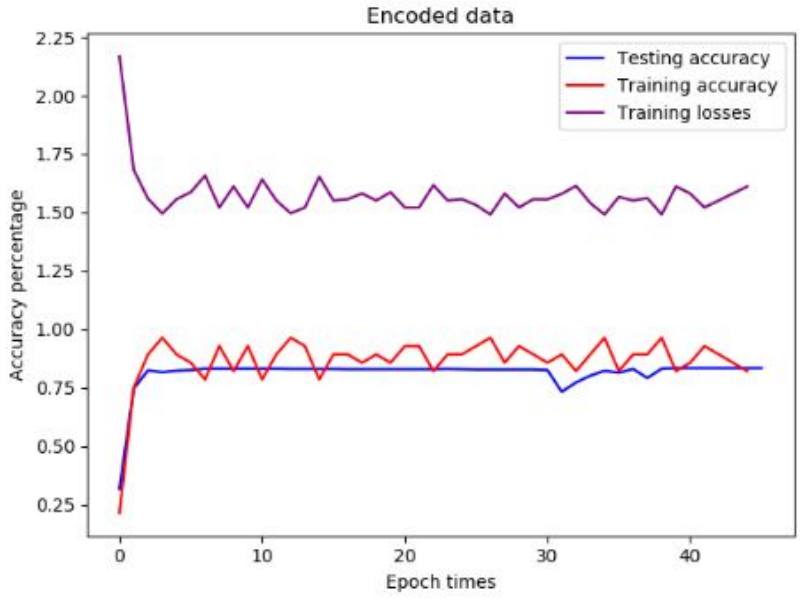


**Fig. 9.** The accuracy during training encoded data process

Compared with the figure 6 and figure 9, the encoded data training process is demonstrated with more losses and less accuracy. Also, the encoded time is countable.

| Epoch | 1 | 51 | 101 | 151 | 201 | 251 | 301 | 351 | 401 | 451 | 501 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Loss | 0.210% | 0.097 % | 0.070% | 0.066% | 0.054% | 0.047% | 0.044 % | 0.038% | 0.043% | 0.036% | 0.045% |

**Table 3.** The loss of the compression data in the 600 epoch.

**Fig. 10** Example of real image: origin image, encoded image, decoded image (from left to right)

From the left to right, the images show the process of the images changes. The images are made directly to show the handwritten number images and the conditions of the data loss. The encoded number is completely blurry and unrecognizable while the origin is clear and easy to recognized. However, the decoded image is easy to distinguished and with less noise. The result of the accuracy is around 80% which could be acceptable.

Then I can conclude the encoded data can be restored partly to the decoded data with less noise. As though some losses are irreversible, it will not seriously damage the quality of the images. Therefore, the autoencoder help to reduce the dimension as my preprocessed implementation.

### 3.3 Convolutional Neural Network(CNN)

| Test time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Testing Accuracy | 96.86% | 97.24% | 96.72% | 95.98% | 97.37% | 96.99% | 96.86% | 97.11% | 96.36% | 96.83% |
| Time | 204.56 | 132.54 | 126.39 | 118.94 | 132.25 | 127.33 | 131.96 | 134.46 | 135.10 | 138.17 |

**Table 4.** The time and testing accuracy of CNN.

It can be seen that the CNN model average accuracy in table 4 reaches 96.83% which is significantly higher than the 91.25% in table 1. Also, the table tendency of the accuracy is increasing as the figure 4 before.
In conclusion, the CNN could pick-up the “useful” data intelligently to heighten the ability to recognize wholefully rolled around the image and understand information better than simple neural network.

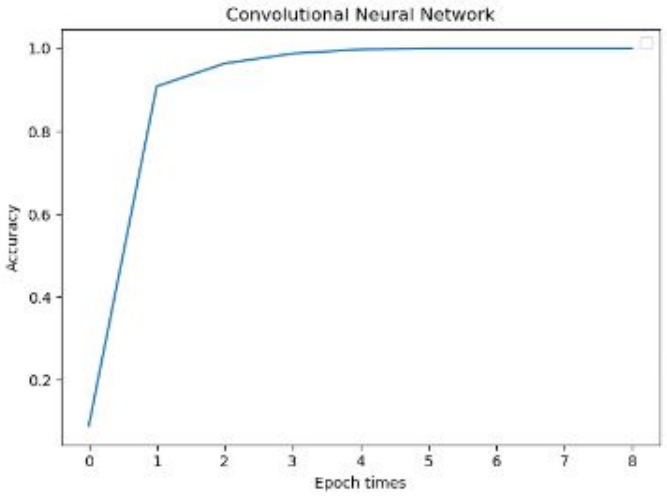


**Fig. 11** Training accuracy of CNN

### 3.4 Encoded data to CNN model

| Test time | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | Average |
|---|---|---|---|---|---|---|---|---|---|---|
| Testing Accuracy | 90.34% | 91.34% | 92.47% | 89.84% | 92.85% | 93.10% | 93.22% | 91.59% | 93.10% | 91.98% |
| Time(s) | 44.46 | 47.62 | 44.68 | 45.83 | 45.60 | 46.08 | 44.40 | 45.99 | 46.94 | 45.74 |

**Table 5.** The testing accuracy and time of encoded data in CNN

Finally, I use the timer to calculate the time of the normal CNN model training and the encoded data training in CNN model. Obviously, the average of 45.74s in table 5 is distinctively shorter than that 138.17s in table 4.

That could be inferred that the encoded data reduce the dimension and decrease the training loads evidently which could increase the efficiency. The CNN model could maintain the data symbols as well as the accuracy (or even have better performance than the simple neural network) and select the symbolized features at the same time.

## 4 Comparison and Conclusion

### 4.1 Comparison

My original source paper is " MetaNet: The Theory of Independent Judges, in Substance Use & Misuse " and " Image Compression using Shared Weights and Bidirectional Networks ". Compared to the paper using the dataset, it uses four models: the Logicon Projection Network (LP), Backpropagation with Momentum and SoftMax(BP)(The previous method I use, so I compare with this data), Back Propagation with Self Momentum and SoftMax(SQ) and Learning Vector Quantization (LVQ). [12] The back-propagation of my experiment is 91.25% on average which is slightly higher than 88.07%(average) the result in the paper, considering the difference of the layers structure. [12]

Therefore, results on the paper performs worse than me before compression. In accuracy, it performs slightly better than mine after compression, because the loss of data in the original dataset. The loss is great, also partly because of the positivity of the data which offsets with the Gaussian distribution caused by the linear activation in the compression layers of the autoencoder. [13] In addition, this loss of the compression is also irreversible and inevitable.

### 4.2 Conclusion

The normal feedforward neural network could train the semion dataset well. I can conclude that the autoencoder and CNN are both effective ways to train the neural network.
The autoencoder could reduce the dimensions significantly and prune some data controlled by the number of the hidden layers, while the CNN will extract some of the symbolized information to learn in a deeper way.
It could be seen that the selection of the "useful" data is random so the accuracy will be fluctuated and unstable. Implementing both of the methods could keep the accuracy and accelerate the training time to improve the efficiency.

### 4.3 Future work

To realize the further compression procedure, my future work is to try other procedures on quantizing the parameters like the weights (share) and bias in a mathematical way and also moving to other formats of the coding for saving memory to larger extent.
In terms of the CNN model, it requires large amount of CPU computation, so I need to find a way to avoid signal stop of CPU by controlling the parameters including the input image size, hidden layers, stripes, paddings, maps, kernel sizes or other factors etc by calculations. In addition, I can choose some colorized images and processed in CNN model with heights to evaluate the model again. At the same time, it is necessary to try on other deep learning model to optimize this handwritten recognition problem.

## References


1. M Buscema, MetaNet: The Theory of Independent Judges, in Substance Use & Misuse 33(2) pp.455 1998
2. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition[J]. arXiv preprint arXiv:1409.1556, 2014.
3. T. D. Gedeon, J.A. Catalan and J. Jin: Image Compression using Shared Weights and Bidirectional Networks pp.3 1996
4. Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks[J]. Chemometrics and intelligent laboratory systems, 1997, 39(1): 43-62.
5. Svozil D, Kvasnicka V, Pospichal J. Introduction to multi-layer feed-forward neural networks[J]. Chemometrics and intelligent laboratory systems, 1997, 39(1): 43-62.
6. En.wikipedia.org. (2018). *Backpropagation*. [online] Available at: https://en.wikipedia.org/wiki/Backpropagation [Accessed 30 May 2018].
7. De Boer P T, Kroese D P, Mannor S, et al. A tutorial on the cross-entropy method[J]. Annals of operations research, 2005, 134(1): 19-67.
8. Hinton G E, Salakhutdinov R R. Reducing the dimensionality of data with neural networks[J]. science, 2006, 313(5786): 504-507.
9. M Buscema, MetaNet: The Theory of Independent Judges, in Substance Use & Misuse 33(2) pp.446 1998
10. Deng L, Seltzer M L, Yu D, et al. Binary coding of speech spectrograms using a deep auto-encoder[C] Eleventh Annual Conference of the International Speech Communication Association. 2010.
11. Xu Z, Wang R, Zhang H, et al. Building extraction from high-resolution SAR imagery based on deep neural networks[J]. Remote Sensing Letters, 2017, 8(9): 888-896.
12. M Buscema, MetaNet: The Theory of Independent Judges, in Substance Use & Misuse 33(2) pp.446 1998
13. Srivastava N, Hinton G, Krizhevsky A, et al. Dropout: A simple way to prevent neural networks from overfitting[J]. The Journal of Machine Learning Research, 2014, 15(1): 1929-1958.