The effect of an auto-encoder on the accuracy of a convolutional neural network classification task

JIAYI CHEN

Research School of Computer Science, Australian National University

Email: isaacchenoc@gmail.com

Abstract. This report aims to discuss how much an auto-encoder affects the accuracy of a convolutional neural network (CNN) classification task in terms of its structure, its training epoch, and its encoded size. Two different auto-encoders are implemented and trained by MNIST dataset. By setting different epoch number and encoded size in training, different restructured data are produced. Original data and the restructured data are used to train a CNN independently. By comparing the classification accuracies, it has been found that increase of epoch number or encoded size lead to increase of classification accuracy, but due to data loss from encoding and decoding the increase stops as a certain epoch number or an encoded size is reached. It also has been found that the structure of an auto-encoder can significantly affect the accuracy of the classification task, because every type of layer has its own unique impact on the auto-encoder.

Keywords: Convolutional Neural Network, Auto-encoder, Classification Problem, MNIST, Deep Learning

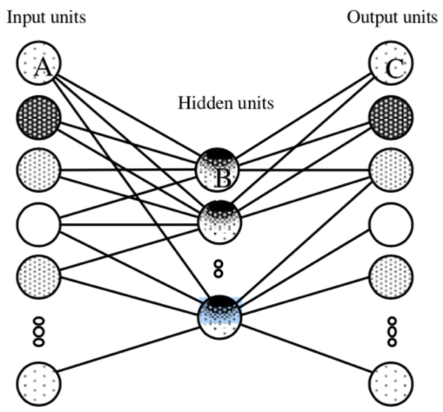
1 Introduction

Our world is a place filled with data, and it is generated from people's lives and benefit people in many ways. One of the most important use of data in people's lives is for prediction. Since in many cases data is following a certain pattern, and by investigating in these patterns, people can make a prediction with a certain degree of accuracy. Today, computer science is much more advanced, people are using machines to predict the result and neural networks (NN) as one of the various approaches that people develop to address this problem. In a NN, a set of data would be used to train the network and can be used to test the performance of the network. Once the training and testing is completed, the network can be used for prediction. Moreover, people's live are often concerned with classification, that is to group a set data based on the attributes of the data. If the classification is all done by human, it would be time consuming and expensive. Hence, NNs are a powerful tool for people to deal with classification problems.

However, since the data in people's life are getting more and more complicative and the classification problems are also getting more and more complex, traditional NNs have become not powerful enough to make correct prediction or classification. For example, traditional NNs are more often dealing with 2D data such as plain text or plain data. When it gets to some complicate data such as images, traditional NNs would become really inefficient. Thus, NNs have been further developed to many sorts of deep NN, where the hidden layers of the NNs become far more complicative. Convolutional neural network (CNN) is a typical deep neural network, which is mainly used to classify images based on the features of the images (DL4J) and it is the focus of this report.

As the human generated data set grows continuously, it is now rapidly increasing in size and is also consuming more physical storage space. As a result, people now often compress the data into a smaller size for storage and decompress them when people need them. There are many sorts of ways to compress and decompress data and NNs can also be used in this case. The NNs which are used to encode and decode a data set are called Auto-encoder. The mechanism that enable compression and decompression is that if the input and the output are the same data set and the number of neurons in the hidden layer is less than input size and output size, when the network is running the input data would be automatically squeezed into the hidden layer and then the data in hidden layer would automatically be expanded and output to the output layer. In this case, the squeezing is actually encoding and the expanding is actually decoding. In this case, the output of the hidden layer is the encoded data and the output of output layer is the reconstructed data. The mechanism of auto-encoder is inspired from the paper *Image Compression using Shared Weights and Bidirectional Networks* (Gedeon, Catalan, & Jin). The paper mainly compares the auto-encoded with share-weights with the auto-encoder with bidirectional network, but the essential mechanism for compression and decompression for both auto-encoders are the same, which is mentioned above.

Figure 1 Auto-encoder topology (Gedeon, Catalan, & Jin)



However, compression and decompression of the data might cause a certain loss of the data. In the case of classification problem, a NN can produce a lower accuracy on classifying the processed data than the original data. However, how much compression and decompression affect the accuracy is uncertain. Hence, this report aims to discuss how much an auto-encoder affects the classification accuracy in terms of the structure of the auto-encoder, number of training epoch of the auto-encoder, and encoded size of the auto-encoder. The famous data set, MNIST (LeCun, Cortes, & Burges), which is a collection of hand written digits, is chosen in this case, since it is a typical 28x28 image data set for CNN classification and it can typically demonstrate how compression and decompression of data can affect the accuracy of CNN classification. Moreover, the classification problem for CNN is to use 28x28 image data to classify what digit the image is.

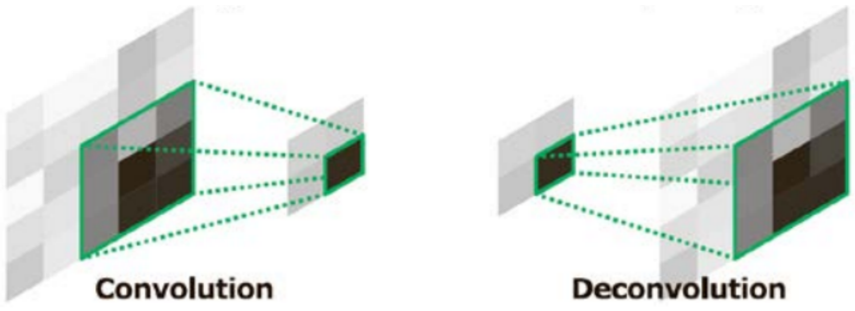
2 Method

2.1 Implementation of Convolutional Auto-Encoder

Since a CNN are a type of NN, the essential mechanism of auto-encoder is the same as what it has been mentioned above. In order to how much the structure of the auto-encoder would affect the classification accuracy, a number different auto-encoders are implemented. However, this report only focuses on two main types of CNN auto-encoder. One is the CNN auto-encoder with only convolutional layers and the other is the CNN auto-encoder with convolutional layers, pooling layers, flatter and full-connection layers.

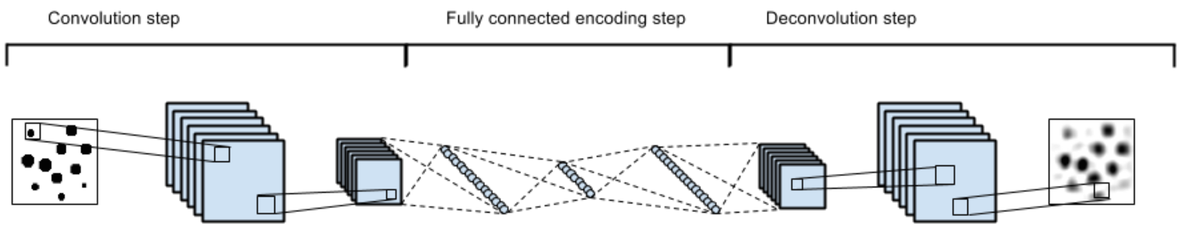
The first type of CNN auto-encoder is implemented with only convolutional layers. There are pairs of convolutional layers in the CNN, where one of the pair would increase the channel of the input data and decrease the high and width of the input data, and the other would decrease the channel of the input data and increase the high and width of the input data. By increasing and decreasing the channel, high and width of the input data, the data is compressed and decompressed in the CNN. It is worth mentioning that during the encoding and decoding, which also are considered convolution and deconvolution (Turchenko, Chalmers, & Luczak), the input image data are remained 4D.

Figure 2 Convolution and Deconvolution (Turchenko, Chalmers, & Luczak)



The second type of CNN auto-encoder is implemented with convolutional layers, pooling layers, flatter and full connection layers, where the convolutional layers are used to increase the channel of the data, pooling layers are used to get the most important part of the image data and reduce its size, flatters are used to swap the data between 2D and 4D, and full connection layers are used to connect all neurons in one layer with all neurons in another layer, which is same as traditional neural networks. In this type of auto-encoder, the input data are encoded through the convolutional layers and pooling layers in a similar way with the first type of auto-encoder. Moreover, the data is flattened into 2D data by the flatter and further encoded in the full connection layers. The data then is decoded in the full connection layers and reshape into 4D data. It is also possible to use convolutional layers to decode the data as what the first type of auto-encoder does and the whole auto-encoder would look like the one shown in figure 3, but for my model, I have simplified the auto-encoder to not include convolutional layers to decode the data.

Figure 3 Convolutional Auto-Encoder (Mugan)



It is worth mentioning that RRELU is used as the activation function in the auto-encoders rather than using RELU. Given that the if the input of RELU is less than zero its output is always zero, it is not ideal to use RELU in an auto-encoder. Since all negative inputs would be mapped to zero by RELU, when it gets to decoding the auto-encoder is not able to map the zeros back to the previous values. Hence, it is better to use an activation function where a unique input is mapped to a unique output and RRELU is an activation function that is able to do so.

$$\boldsymbol{RELU}\text{:}\ \ \boldsymbol{f(x)} = \begin{cases} \boldsymbol{0} & \boldsymbol{for\ x < 0} \\ \boldsymbol{x} & \boldsymbol{for\ x \geq 0} \end{cases}$$

$$\boldsymbol{RRELU}\text{:}\ \ \boldsymbol{f(a,x)} = \begin{cases} \boldsymbol{ax} & \boldsymbol{for\ x < 0} \\ \boldsymbol{x} & \boldsymbol{for\ x \geq 0} \end{cases}$$

2.2 Training using original data and processed data

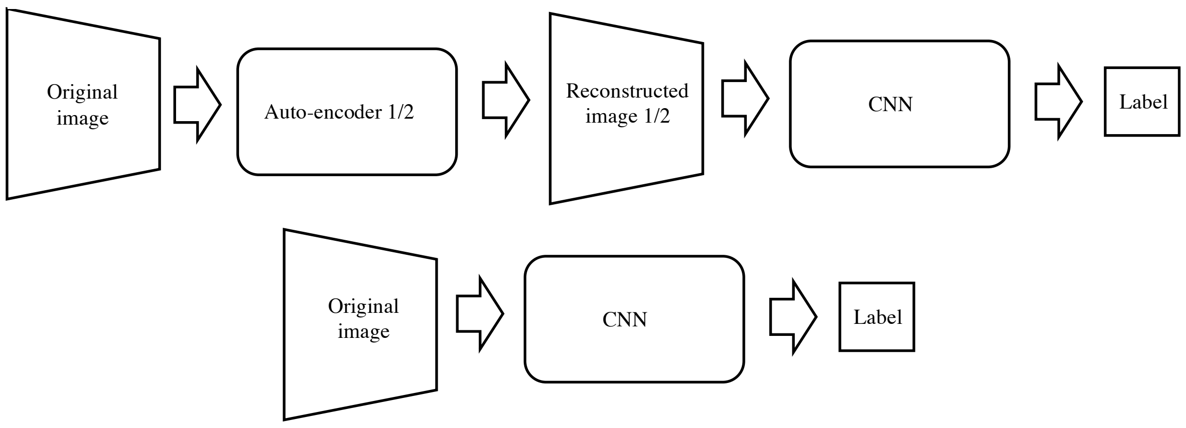
It is worth mentioning that initially this report was focusing on taking the encoded data as the input of the next CNN. However, this focus was abandoned because I realized that people usually would not use the encoded data directly and this kind of data is mainly for storage. What people are really interested and use directly is the data which have been encoded before but now is decoded. Hence, the report now focusing on taking the reconstructed data as the input of next CNN.

After the auto-encoders are implemented, the MNIST data set is used to train two auto-encoder independently. The epoch size, which stands for how many times the auto-encoder is trained, and the encoded size, which stands for the size of the compression of the image data, are tuned in both of the auto-encoders, in order to see how these two factors affect the later classification task. When tuning the encoded size for the auto-encoders, the kernel (K), stride(S), padding(P), output size(O) and the input size(W) should always satisfy the following equation, otherwise the encoding and decoding would fail (Deshpande, 2016).

$$\boldsymbol{O} = \frac{\boldsymbol{W - K + 2P}}{\boldsymbol{S}} + \boldsymbol{1}$$

After the training of the auto-encoders, the output of the auto-encoders is the processed data that have been encoded and decoded, and the processed data is taken as an input to train next CNN, which consists of two convolutional layers, two pooling layers, a flatter and two full connection layers, and is used for classifying the label of the data. The original MNIST data is also used to train the CNN directly, in order to see how much the classification accuracy is affected by the auto-encoders. It is worth mentioning that the setting for the CNN always remain the same, since the effect of the auto-encoder is the only interest of this report. Moreover, each time only one of the two factors is changed, so that two factors would not interfere with each other.

Figure 4 Current overall process



2.3 Accuracy comparison among processed data and original data

Both of the auto-encoders run in different setting and for each setting the auto-encoders run ten times. The outputs of the auto-encoders are then passed to the next CNN for classification. After the CNN is trained, it is tested in order to see how accurate the classification is, where the accuracy is calculated using the formula:

$$\boldsymbol{Accuracy = \frac{correct\ classification\ \times\ 100\%}{total\ classification}}$$

Furthermore, as each setting is run ten times for both of the auto-encoders, the average accuracy is also calculated.

Since a reconstructed image which is similar with the original image should indicate an auto-encoder with good performance, the original image data and restructured image data are all converted to actual images. By comparing the decoded image with the original image, we should basically know that how much the classification accuracy is affected by encoding and decoding.

3 Results and Discussion

3.1 Classification accuracy for the auto-encoders with different epoch numbers

The Classification accuracies for the auto-encoders with different epoch numbers are shown on the figure 5 and 6.

Figure 5 Accuracy box plot - AE 1 (with fixed encoded size 16*16)

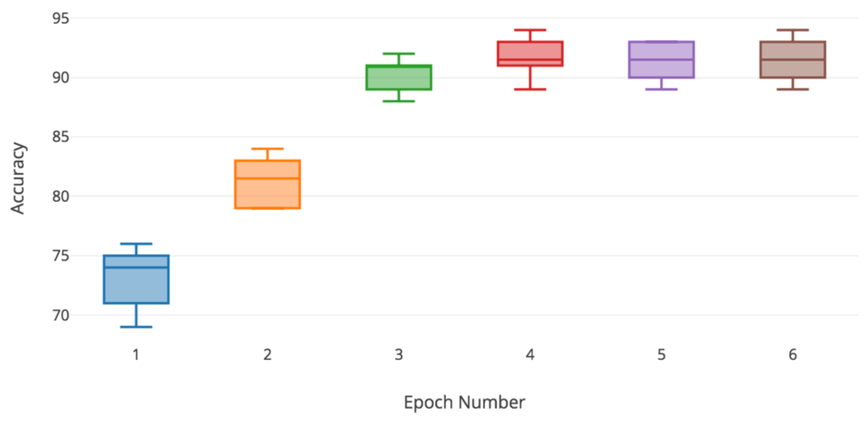
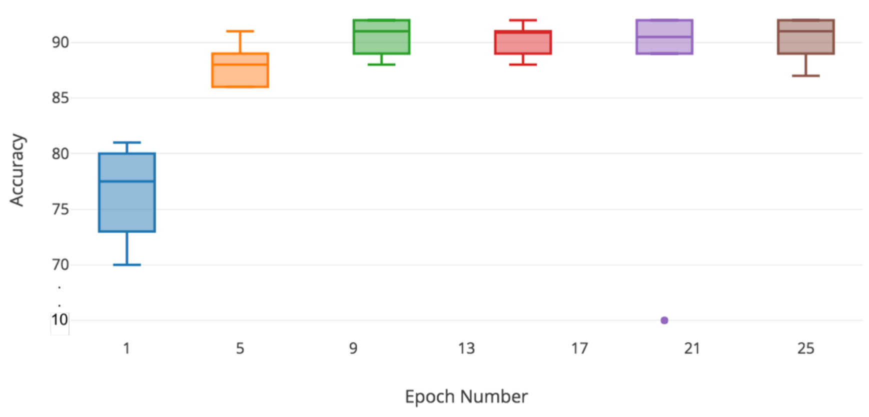
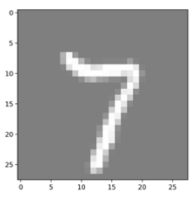
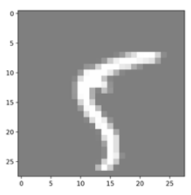
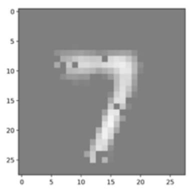
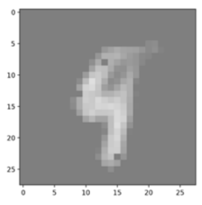


Figure 6 Accuracy box plot - AE 2 (with fixed encoded size 24)



As the epoch number for training the auto-encoders increases, the classification accuracy for the CNN also increases. When the epoch number is small, the accuracy increases rapidly, and when it gets to a certain point, the accuracy basically stops increasing. This is reasonable because as the auto-encoders are training, the output of them becomes more and more similar with the target, which in the case of auto-encoders is the input data. Hence, the accuracy increases as the epoch size increases. However, since encoding and decoding cause certain loss of the data, even if the epoch size is large enough, the accuracy is still less than the original accuracy. Hence, the accuracy in the box plots stops increasing at a certain point. When the auto-encoders are trained with a sufficient number of epochs, the accuracy for both cases above is about 92%. Compared to the CNN which takes original data as input and achieves the accuracy of 99%, the accuracy loss due to auto-encoders is not too much. The original images and restructured images for different training epoch are shown in table 1, where a larger training epoch leads to a clearer restructured image, which means a higher classification accuracy.

Table 1 Image comparison (different epoch)

	10 Epoch	1 Epoch
Original		
Restructured		

Thus, given an auto-encoder with fixed encoded size, an increasing number of epoch leads to an increasing classification accuracy, but even if the epoch number is large enough the classification accuracy is still less than the original classification accuracy, due to the loss from encoding and decoding.

3.2 Classification accuracy for the auto-encoders with different encoded size

The Classification accuracies for the auto-encoders with different encoded size are shown on the figure 7 and 8.

Figure 7 Accuracy box plot - AE 1 (with fixed epoch number 1)

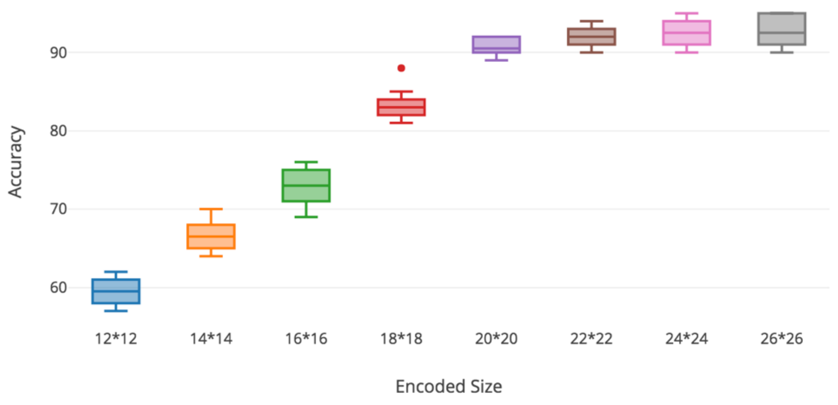
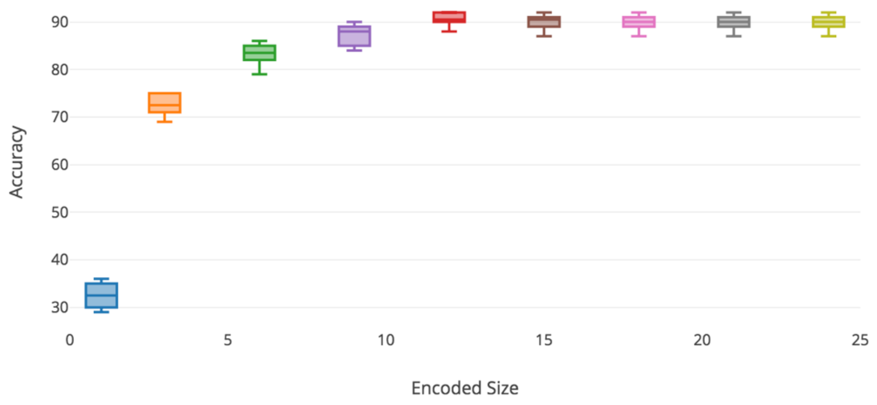
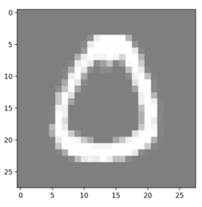
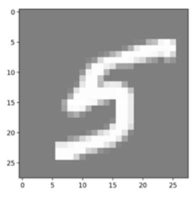
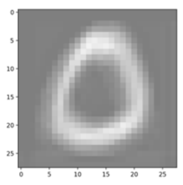
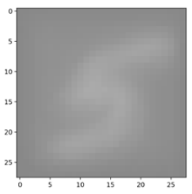


Figure 8 Accuracy box plot - AE 2 (with fixed epoch number 10)



It is reasonable to see that a lower encoded size leads to a lower accuracy and the accuracy becomes stable when the encoded size reaches a certain number, because a lower encoded size means more loss in the image data, which means the restructured image would be more different from the original data. Moreover, even though the encoded size is large enough, there is still loss in the image data. Hence, the accuracy is still less than the original accuracy and the accuracy in the box plot stops increasing. When the auto-encoders are trained with the encoded size which is large enough, the accuracy for both cases above is about 92%. Compared to the CNN which takes original data as input and achieves the accuracy of 99%, the accuracy loss due to auto-encoders is not too much. The original images and restructured images for different encoded size are shown in table 2, where a smaller encoded size leads to a worse restructured image, which means a lower classification accuracy.

Table 2 Image comparison (different encoded size)

	Encoded size 18 * 18	Encoded size 12 * 12
Original		
Restructured		

Thus, given an auto-encoder with fixed epoch number, an increasing encoded size leads to an increasing classification accuracy, but even if the encoded size is large enough the classification accuracy is still less than the original classification accuracy, due to the loss from encoding and decoding.

3.3 Classification accuracy for different auto-encoders

As the graphs above show, although AE 1 and AE 2 are both auto-encoders, their effect on the given classification task are quite different. Specifically, AE 2 allows the image data to be encoded to a much smaller size than AE 1 does, while their leading classification accuracies are the same. For example, although the image data that are processed by both of the auto-encoders can result in an accuracy of 90%, the encoded size for AE 1 is 20*20 but the encoded size for AE 2 is just 12. Moreover, the time taken to train AE 2 is much shorter than the time taken to train AE 1, which presumably is because the computation in AE 1 is in 4D and the computation in AE 2 is in 2D as there is a flatter in AE 2. Hence, AE 2 does a much better job than AE 1.

In the paper *A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe* (Turchenko, Chalmers, & Luczak), an auto-encoder with convolutional layers, pooling layers, switches, deconvolutional layers and unpooling layers is proposed. Moreover, compared to AE 1 and AE 2, the proposed auto-encoder does a better job as the data which is processed by it leads to the classification accuracy that is at least 88% on MNIST in all encoded sizes. In particular, when the encoded size is 2, the propose auto-encoder can result in an accuracy of 88%, while AE 2 is only about 35%. There is such a difference presumably because AE 2 has no unpooling or deconvolutional layers to restructure the lost data due to pooling and convolutional layers, which the proposed auto-encoder does.

AE 2 does a better job compared to AE 1, since it has flatter and full connection layers to simplify the computation. The proposed auto-encoder from the paper mentioned above does a better job compared to AE 2, since it has deconvolutional and pooling layers to restructure the lost data. Hence, the structure of the auto-encoder does significantly affect the classification accuracy, as different types of layer have different impact on the encoding and decoding of the data.

4 Conclusion and Future Work

Two auto-encoders are implemented in different ways. One consists of convolutional layers only and the other consists of convolutional layers, pooling layers and full connection layers. MNIST dataset is used to train the auto-encoders and produce the restructured data. Different epoch numbers and encoded sizes are set to train the auto-encoders. The original data and the restructured data are used independently to train a CNN which is meant to classify the label of MNIST in order to see how much the auto-encoders affect the classification accuracy. It has been found that an increasing epoch number or encoded size leads to an increasing classification accuracy. However, due to loss on original data in encoding and decoding, the accuracy becomes stable as a certain epoch number or encoded size is reached. It also has been found that the structure of the auto-encoder affects the classification accuracy significantly as each kind of layer has its own unique impact on the auto-encoder.

In the future, there is certain work to do in order to further study the impact of auto-encoder on the classification accuracy. Firstly, as figure 6 shows, outliers appear on the box plots. Hence, it is necessary to further study why it is caused and how to avoid it. Secondly, a different activation function can lead to a completely different result, such as the case that if RELU is used in AE 1 or AE 2, the classification accuracy would be about 10%. Hence, it is necessary to further study about which activation function is the most suitable for AE 1 and AE 2. Thirdly, since epoch number and encoded size are the focus of this report, other settings of an auto-encoder have not been studied sufficiently, such as how much the channel of an auto-encoder affect its performance. It is necessary to study other settings in order to further develop AE 1 or AE 2.

Acknowledgments. (1) The code, including how to build, train and test a CNN, was written with the help of the code from CNN Labs by Professor Tom Gedeon, ANU. (2) The main technique of the code: Autoencoder was inspired by the paper: Image Compression using Shared Weights and Bidirectional Networks (Gedeon, Catalan, & Jin). (3) Thanks to Xuanyu Chen, I have become clear about how to implement a CNN autoencoder, that is to build a CNN, where convolutional layers and full connectional layers are opposite, the dimension of convolutional layers should increase then decrease, and dimension of full connection layers should decrease then increase. (4) Thanks to Ao Feng, I have become clear about how to use view function to flatten or reshape the CNN in the Auto-encoder, which is swapping in 4D and 2D. Moreover, the size of the input data would affect the parameter that view function takes, so it's better to include it in the definition of Auto-encoder.

References

Deshpande, A. (2016). *A Beginner's Guide To Understanding Convolutional Neural Networks Part 2*. Retrieved from Adit Deshpande: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks-Part-2/

DL4J. (2016). *A Beginner's Guide to Deep Convolutional Neural Networks (CNNs)*. Retrieved from DL4J: https://deeplearning4j.org/convolutionalnetwork#

Gedeon, T. D., Catalan, J., & Jin, J. (n.d.). Image Compression using Shared Weights and Bidirectional Networks. *Proceedings 2nd International ICSC Symposium on Soft Computing (SOCO'97)*, 374-381.

LeCun, Y., Cortes, C., & Burges, C. J. (n.d.). *THE MNIST DATABASE* . Retrieved from Yann LeCun: http://yann.lecun.com/exdb/mnist/

Mugan, J. (2015). *Convolutional autoencoders in python/theano/lasagne*. Retrieved from Swarbrickjones.wordpress: https://swarbrickjones.wordpress.com/2015/04/29/convolutional-autoencoders-in-pythontheanolasagne/

Turchenko, V., Chalmers, E., & Luczak, A. (n.d.). *A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe*. Lethbridge: University of Lethbridge.