*Machine Learning and Artificial Intelligence*
*J.-L. Kim (Ed.)*


*doi:10.3233/FAIA230761*

# Compression and Decompression Using Deep Neural Network

Jinxin WEI[a1], Zhe HOU[b]
[a] *Vocational School of Juancheng, Juancheng 274600, China*
[b] *No.1 High School of Juancheng, Juancheng 274600, China*

**Abstract.** An auto-encoder which can be split into two parts is designed. The two parts can work well separately. The top half is an abstract network which is trained by supervised learning and can be used to classify and regress. The bottom half is a concrete network which is accomplished by inverse function and trained by self-supervised learning. It can generate the input of abstract network from concept or label. It is tested by tensorflow and mnist dataset. The abstract network is like LeNet-5. The concrete network is the inverse of the abstract network. A picture can change to label that is compression, then change it back from label that is decompression. So compression and decompression can be realized by the autoencoder. Through test, the absolute function can do generation task well, while the leaky relu function can do classification task well. Lossy compression can be achieved by abstract network and concrete network with absolute function. With one-hot encoding, the compression ratio is 5.1% when decompression quality is good. With binary encoding, the compression ratio is 2% when decompression quality is ok. The mean PNSR of five pictures is 19.48dB. When jump connection and negative feedback are used, the decompression performance is good. The mean PNSR of five pictures is 29.86dB. The compression ratio is 56.1%.

**Keywords.** Compression, decompression, deep neural network, regression, jump connection, auto-encoder, absolute function, negative feedback, binary encoding, machine learning

## 1. Introduction

In the realm of image processing and computer vision (CV), machine learning (ML) architectures are widely applied. Convolutional neural networks (CNNs) solve a wide range of image processing issues and can solve image compression problem. Compression of images is necessary due to bandwidth and memory constraints. There are ML architectures that lossy image compression used including different auto-encoders (AEs) such as convolutional auto-encoders (CAEs), variational auto-encoders (VAEs), recurrent neural networks (RNNs), CNNs, generative adversarial networks (GANs), principal component analysis (PCA) and fuzzy means clustering [1]. The methods change the picture into bottleneck or latent space or features or low dimension data, then change it back.

In another paper [2], an autoencoder which can be split into two parts is designed. The two parts can work well separately. The top half is an abstract network which is

[1] Corresponding Author: Jinxin WEI, Vocational School of Juancheng, Juancheng, China. E-mail: wjxabai@163.com.

trained by supervised learning and can be used to classify and regress. The bottom half is a concrete network which is accomplished by inverse function and trained by self-supervised learning. It can generate the input of abstract network from concept or label. The input can change to any form by encoder and then change it back by decoder through inverse function. A picture can change to label that is compression, then change it back from label that is decompression. So, compression and decompression can be realized by the autoencoder.

The abstract network is like LeNet-5. The abstract network is 3 layers convolutional neural network and 2 layers fully connected network, the concrete network is the inverse of the abstract network. The architecture is shown in Figure 1.

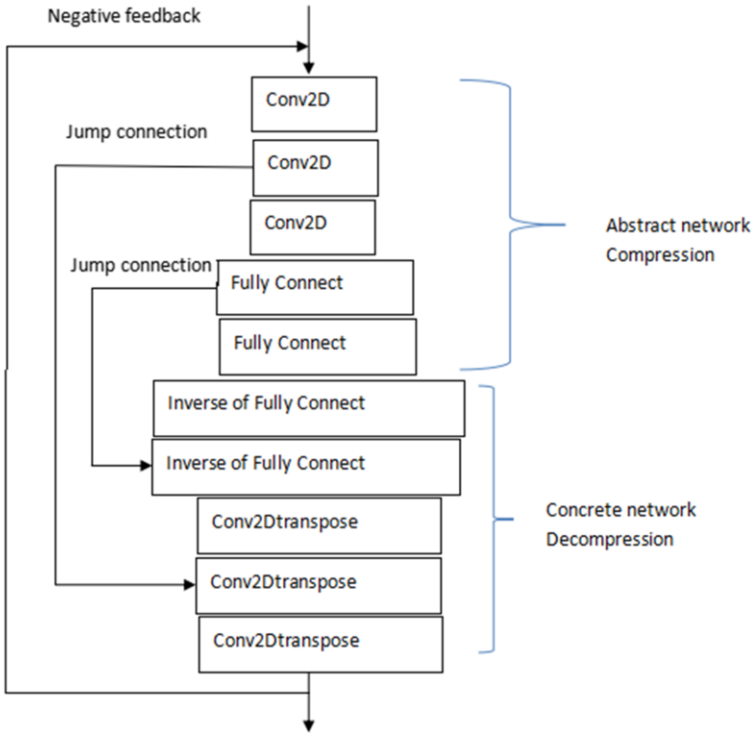


**Figure 1.** The Architecture of Network.

## 2. The inverse function

The following are inversions of the functions. The function of fully connected layer is

$$y = wx + b \quad [3] \tag{1}$$

so, the inverse function is

$$y = w^{-1}x - w^{-1}b \tag{2}$$

Because the linear function's inverse function is also linear, so the fully connected layer's structure of concrete network is the same as abstract network layer's. Because w is a matrix, we need the inverse of w, so the w needs to be square matrix. But the real situation is that the dimension of w is determined by the neuron numbers of the two layers next to each other, so this network can not reproduce the inputs, but approximate the inputs.

The function of leaky relu is

$$y = \max(x, \alpha x) \text{ [4]} \tag{3}$$

The inverse function is

$$y = \min(x, \frac{1}{\alpha} x) \tag{4}$$

The function of absolute [5] is

$$y=|x| \tag{5}$$

The inverse function is

$$y=|x| \tag{6}$$

Because the inverse of the softmax function which used by classification is a approximation method [2], it can change the distribution of the probability, so don't use softmax in order to generate the input well. You can read my another paper [2] for the detail. Most deep learning frameworks all have the transpose of convolution function, just use conv2Dtranspose layer. The input shape of Conv2d and the output shape of Conv2dtranspose are the same. The output shape of Conv2d and the input shape of Conv2dtranspose are the same. Conv2d is dimension reduction, Conv2dtranspose is rise-dimension.

### *2.1. The test on leaky relu and absolute function*

The test is on mnist dataset and autoencoder which architecture is shown in Figure 1. Activation function is leaky relu and absolute function [5], no padding and no maxpooling. Loss function is mse [3], optimizer is adam [3]. We train the top half first, then set it as untrainable, then train the bottom half. The results are the following pictures.

In Figure 2-Figure 5, the left image is input, the label is argmax function of the predict of output of abstract network, the right image is the output of concrete network. In Figure 2, when epoch=1, the labels of input are all right. But the image of output is very blurred. Meanwhile, in Figure 3, the labels of input are not all right. But the image of output is clear. One conclusion can easily draw. It is that the leaky relu function can do classification task well, while the absolute function can do generation task well. In Figure 4 and Figure 5, epoch=30, the result is almost the same between the two activation function. One different is the background color of absolute function is whiter than leaky relu function. So the absolute function can do generation task well.

The small compression ratio is achieved. Such as one mnist picture is 28*28 byte, the data type of label is float32, so it is 32*10 bits. For example, the label is 0100000000 which represents 1. The compression ratio is (320/784*8)*100%=5.1%.

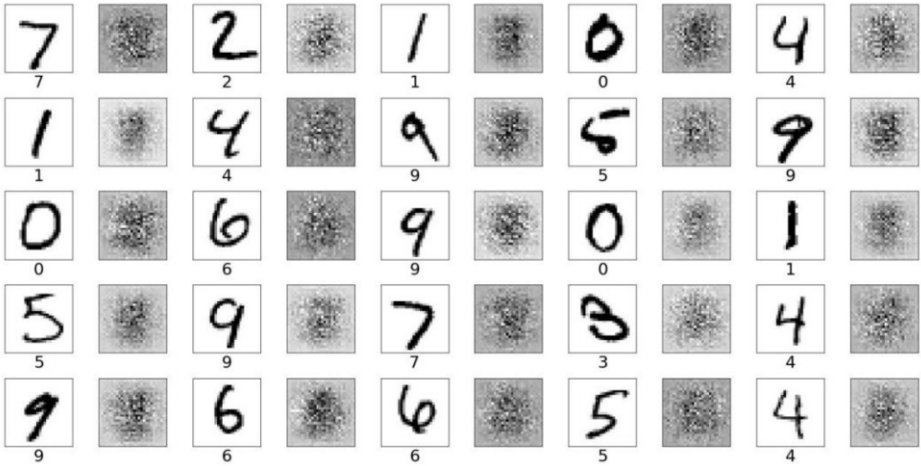

**Figure 2.** Leaky Relu Function, Epoch=1.

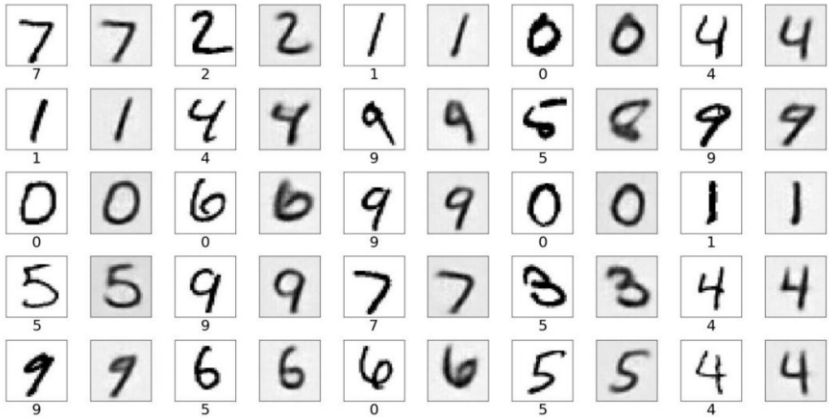

**Figure 3.** Absolute Function, Epoch=1.

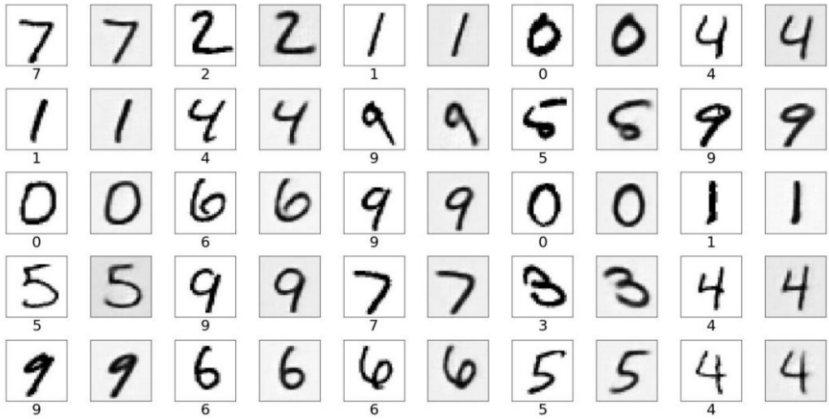

**Figure 4.** Leaky Relu Function, Epoch=30.

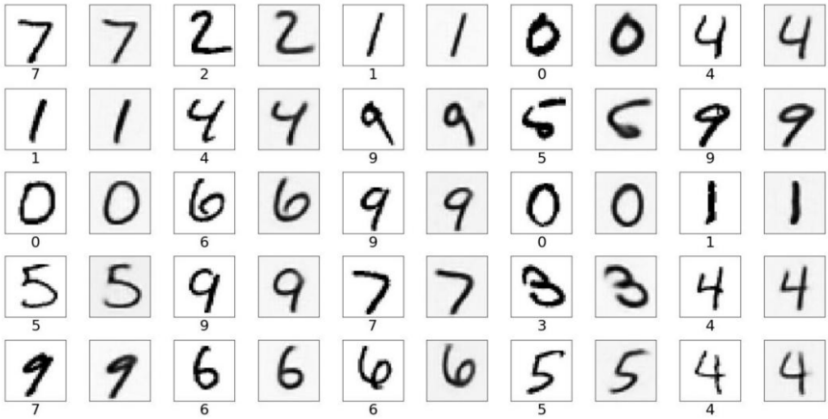

**Figure 5**. Absolute Function, Epoch=30.

## *2.2. The test on binary encoding*

In another paper [6], one new encoding is proposed. When there are 10 classes which are used by mnist dataset, the encode is 0000, 1000, 0100, 1100, 0010, 1010, 0110, 1110, 0001, 1001. There are 4 bits for binary encoding method vs 10 bits for one-hot encoding. In Figure 6-Figure 7, the left image is input, the label is round function of the predict of output of abstract network, the right image is the output of concrete network. When binary encoding is used, the compression ratio is smaller than one-hot encoding. The compression ratio is 32*4/(784*8)*100%=2%. What about the quality of decompression? let's see Figure 6. In Figure 6, the quality of decompression is worse than one-hot encoding, but it is ok. The mean PNSR of five pictures is 19.48dB. The mean PNSR of ten pictures is 18dB.

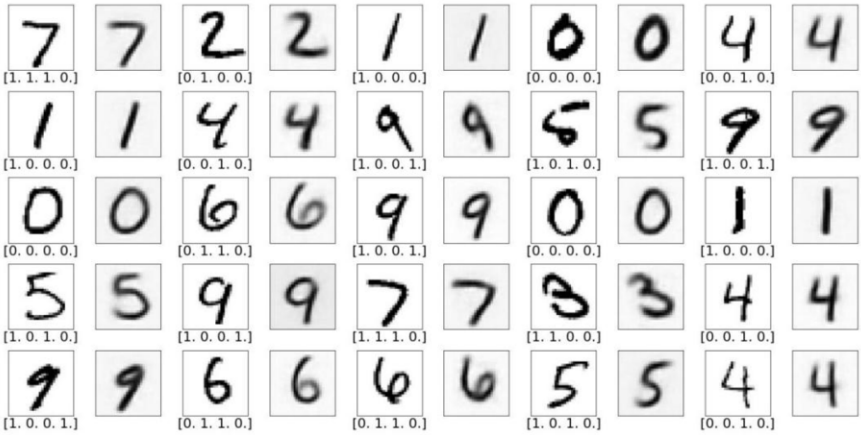

**Figure 6**. Absolute Function, Binary Encoding.

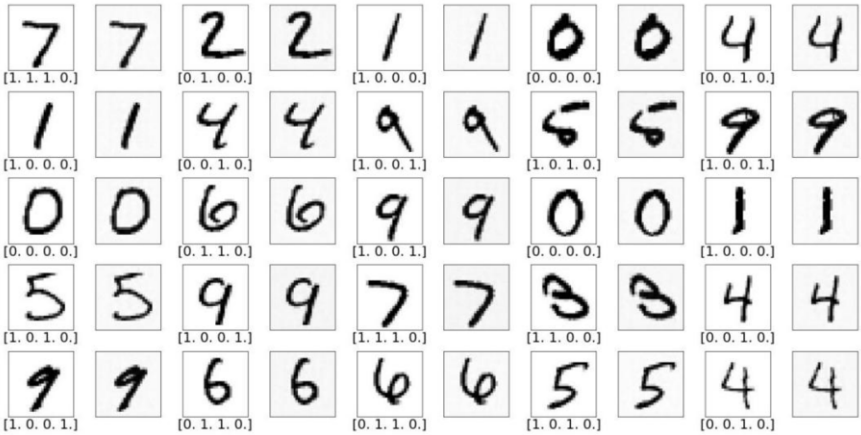

**Figure 7**. Absolute Function with Jump Connection and Negative Feedback, Binary Encoding.

### *2.3. The test on jump connection and negative feedback*

In order to generate the output which is very similar to input, jump connection and negative feedback (which are in Figure 1) can be used. The output of one layer of abstract network which can be seen as the knowledge of features it has learned before is connected to the input of the symmetrical layer of the concrete network, then take the mean as the new input. If the output of layer two is B, the input of layer nine is $B \pm \xi$, $\xi$ is the error because the inverse function which used before is the approximation function. So $\frac{1}{2}(B + B \pm \xi) = B \pm \frac{1}{2}\xi$, the error decreases. The more jump connection (more knowledge about features), the less training time, the more similarity. It fits the process of learning. Figure 7 shows the result of jump connection (layer 2 and layer 9) and negative feedback. It shows that the background is no longer dark when negative feedback and jump connection are used, the output is very similar with input. Why is negative feedback? Inspired by principle of automatic control, negative feedback is added. Because the whole network is like the proportional integral differential parts of automatic control, our aim is to make output very similar to input, So let the subtraction of input and output as the new input will decrease the difference of output and input.

Because the picture is 28*28*8 bits, when the jump connection between two large layers which nodes are larger than 186, the compression is no longer compression. Only when the jump connection between two small layers is suit for compression, so the first fully connected layer which has 100 nodes is suit for compression. The compression ratio is (32*10+100*32)/(784*8)*100%=56.1%. The decompression performance is shown in Figure 8. The mean PNSR of five pictures is 29.86dB. The mean PNSR of ten pictures is 29.33dB. In Figure 9[1], it is the PSNR achieved by different models. It shows that the PSNR of different models is around 30dB.

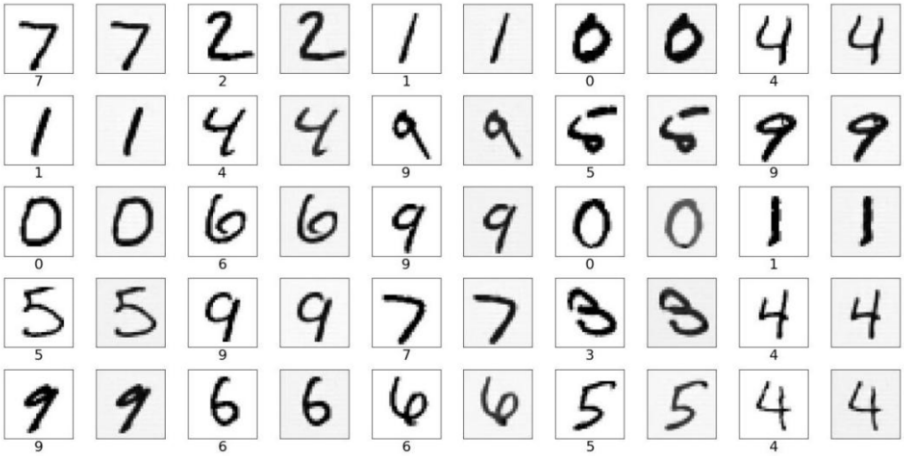

**Figure 8.** Absolute Function with Jump Connection (First Fully Connected Layer) and Negative Feedback, one-hot Encoding.

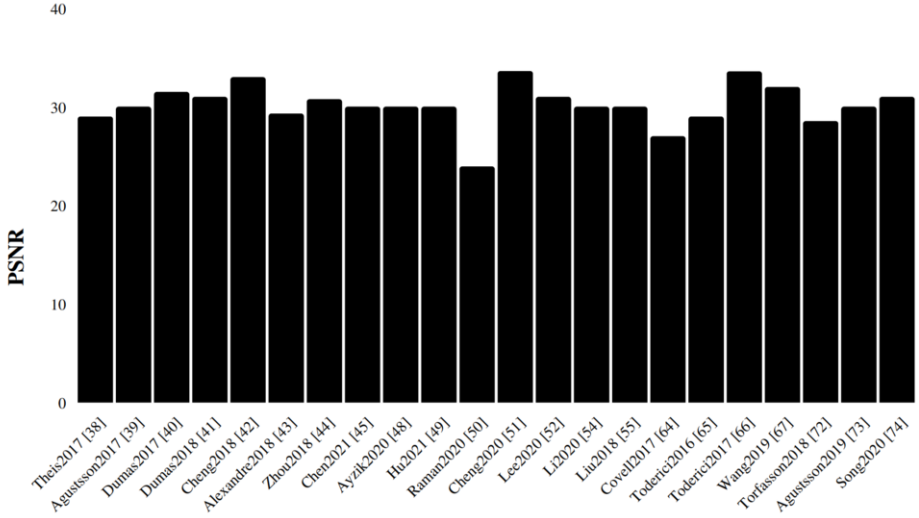


**Figure 9**. PSNR achieved at minimum bpp by different models.

## 3. Conclusion

Through test, the absolute function can do generation task well, while the leaky relu function can do classification task well. Lossy compression can be achieved by abstract network and concrete network with absolute function. With one-hot encoding, the compression ratio is 5.1% when decompression quality is good. With binary encoding, the compression ratio is 2% when decompression quality is ok. The mean PNSR of five pictures is 19.48dB. When jump connection and negative feedback are used, the decompression performance is good. The compression ratio is 56.1%. The mean PNSR of five pictures is 29.86dB.

## References

[1] Sonain J, Md. JP, and MuhibUr R. Learning-driven lossy image compression; a comprehensive survey, Arxiv, 2022
[2] Wei JX, Ren QY. A Functionally Separate Autoencoder. Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1. p. 553-560
[3] Tensorflow Tutorials and Apis, https://tensorflow.google.cn/learn, last accessed 2022/10/20
[4] Ian G, Yoshua B, Aaron C. Deep Learning, MIT Press, 2016. p.192-193
[5] Wei JX, Hou Z. Activation Function: Absolute Function, One Function Behaves more Individualized, TechRxiv, 2021. Preprint. https://doi.org/10.36227/techrxiv.17639525.v3
[6] Wei JX, Hou Z. Binary Encoding for Label, TechRxiv, 2022. Preprint. https://doi.org/10.36227/techrxiv.21353763.v1