A Functionally Separate Autoencoder

Jin-xin Wei, Qun-ying Ren

***Abstract*—According to kids' learning process, an auto-encoder is designed which can be split into two parts. 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. The network can achieve its intended functionality through testing by mnist dataset and convolution neural network. Round function is added between the abstract network and concrete network in order to get the the representative generation of class. The generation ability can be increased by adding jump connection and negative feedback. At last, the characteristics of the network is discussed. The input can be changed to any form by encoder and then change it back by decoder through inverse function. The concrete network can be seen as the memory stored by the parameters. Lethe is that when new knowledge input, the training process makes the parameters change.**

***Index-Terms*—auto-encoder, supervised learning, self-supervised learning, inverse function, abstract network, concrete network, computer vision, jump connection, negative feedback**

ABSTRACT network(namely prediction network,always used as classification and regression) is common now. But the concrete network(namely generation network)which generates concrete information from concept or label is rare. Why we need concrete network? I take the students' learning process as example. An infant teacher takes out several pictures. There is a horse in all the pictures. She points at one of the pictures and says it's a horse, the horse has what features, the features have what shapes. So the kids learn how to recognize a horse. It is supervised learning. Then the teacher asks a question that how to draw the horse? The kids can draw their own horse by using the features they recognized. It is the process from label or concept to concrete information and it's self-supervised learning. I design the network according to this process to

Abstract network(prediction network) is easy to design. How to design the concrete network? The following is the detail. The top half is an abstract network which is trained by supervised learning and the bottom half is a concrete network which is accomplished by inverse function of the top half and trained by self-supervised learning. We train the top half first, then set it as untrainable, then train the bottom half. I take two layers abstract network(binary classification)which is fully connected and two layers concrete network as example. It is shown as Fig. 1

The following are inversion of the functions. The function of fully connected layer is $y = wx + b$, so the inverse function of it is $y = w^{-1}x - w^{-1}b$. 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.

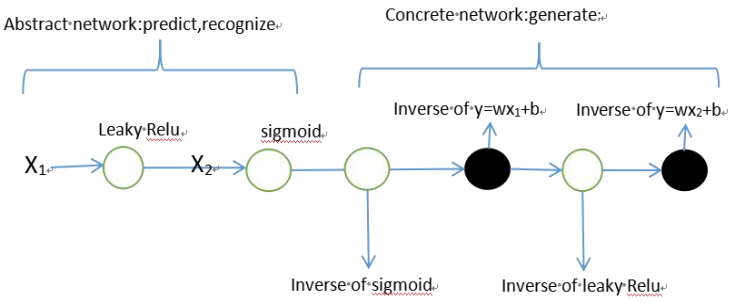

Fig. 1. The Functionally Separate Auto-encoder

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 layer next to each other, so this network can not reproduce the inputs, but approximate the inputs. The activation function sigmoid is $y = \frac{1}{1 + e^{-x}}$ [1]. The inverse function is $y = \ln \frac{x}{1 - x}$. The function of leaky relu is $y = \max(x, \alpha x)$ [2]. The inverse function is $y = \min(x, \frac{1}{\alpha} x)$. Why we use the activation function leaky relu and sigmoid? Because the inverse function of relu function will lose some value when x is negative and this will cause generation loss. When you face multi-class problems, you can use softmax function. It is $y_i = \frac{e^{x_i}}{\sum_1^m e^{x_j}}$ [1]. Because the inverse function of it is difficult to derive, so we can use $y = \ln \frac{x}{1 - x}$ instead. Take y_1 , x_1 as example, the derivation is as follows.

$$y_1 \sum_1^m e^{x_j} = e^{x_1} \quad (1)$$

$$\frac{y_1 (e^{x_2} + \cdots + e^{x_m})}{1 - y_1} = e^{x_1} \quad (2)$$

F. A. Author, Jin-xin Wei is with Vocational School at Juancheng,CHINA (e-mail: wjxabai@163.com).

S. B. Author, Qun-ying Ren.is with Bureau of Emergency Management of Juancheng County, CHINA(e-mail: renqy1987@163.com).

$$\ln\frac{y_1(e^{x_2}+\cdots+e^{x_m})}{1-y_1}=x_1$$ (3) When $e^{x_2}+\cdots+e^{x_m}$ =1, $x_1=\ln\frac{y_1}{1-y_1}$. When y_1 is large($1\ge y_1\ge 0.9$), $y_2\cdots y_m$ is small, so $e^{x_2}\cdots e^{x_m}$ is small,so $e^{x_2}+\cdots+e^{x_m}<1$. Setting it to 1 will increase y_1. When y_1 is small, $e^{x_2}+\cdots+e^{x_m}>1$, setting it to 1 will decrease y_1. So the effect is increasing large probability, decreasing small probability, then the network's robustness increases. It's important to note that y_1 maybe equal to 1 or 0, so $1-y_1$ maybe equal to 0 or x_1 maybe $-\infty$, so we need the function to be $y=\ln\frac{x+1e-7}{1-x+1e-7}$.

Because the generation effect of the network relies on the accuracy of abstract network's classification and convolution neural network's accuracy is 1%-2% larger than fully connected network, so we can use convolution neural network to replace the fully connected network. Most deep learning frameworks all have the transpose of convolution function.

The experiment is done by mnist dataset and tensorflow framework. The abstract network has 3 convolution layers and 2 fully connected layers, no maxpooling and padding. Because they will cause generation loss. The concrete network is the inverse function of abstract network. When training the abstract network, loss function is sparse_categorical_crossentropy[3], optimizer is adam[3], metrics is accuracy[3]. Because the number of layers is small, so the accuracy of the abstract network on test data is 98.3%-99%. This will affect the generation effect, but slightly. When training the concrete network, loss function is mean squared error, optimizer is adam, metrics are mean absolute error and cosine similarity, label is input. The test result on test data is shown on Fig2. We can see that the network generates inputs well although it never know the test data before.

From Fig. 2(a), we can see that the results of these classification are all right, the generation is ok although it is a little blurry. From most of generation outputs, we can see that the similarity of the generation and input is not very high and the generation tend to be the nearest type of input in training dataset. The number 3 and 0 are the best example. Let's look at the second 9. The bottom part of 9 have two inclines, one is vertical, the other is left-leaning. It tells us that the classification result of the abstract network is between the two kinds of 9, so two tails appear. We can conclude that the commonality is clear and the individuality is blurry which reflects the generalization ability of the network.

Because the label of one class is same, when we use label as input of concrete network, the output is the same. To achieve this, we add round function(which is in Fig3) between abstract network and concrete network. The result is shown in Fig. 2(b) We can see that the type of the output is the nearest to all the same class. The output can be seen as 'the mean of the same class' and the representative of class.

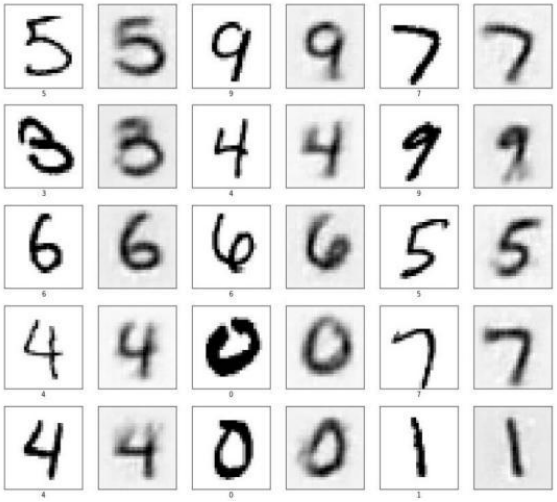

Fig. 2(a). Pure inverse function

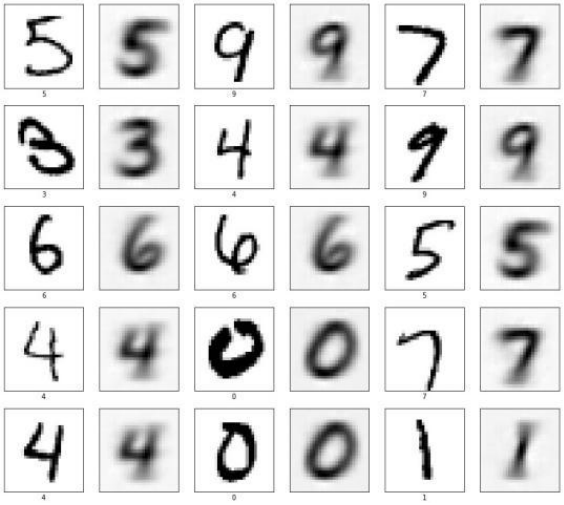

Fig. 2(b). Round function

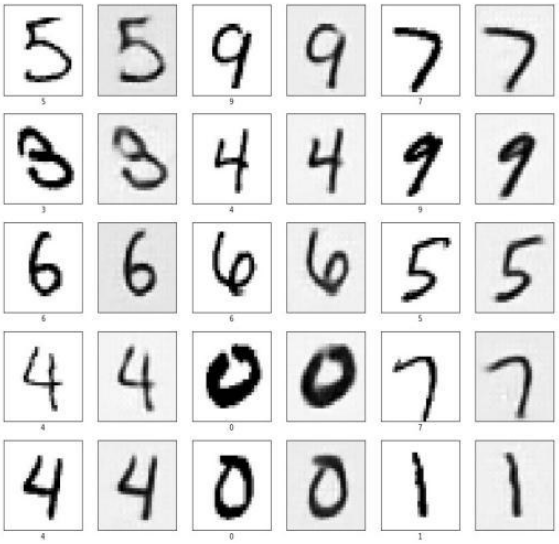

Fig. 2(c). Jump connection

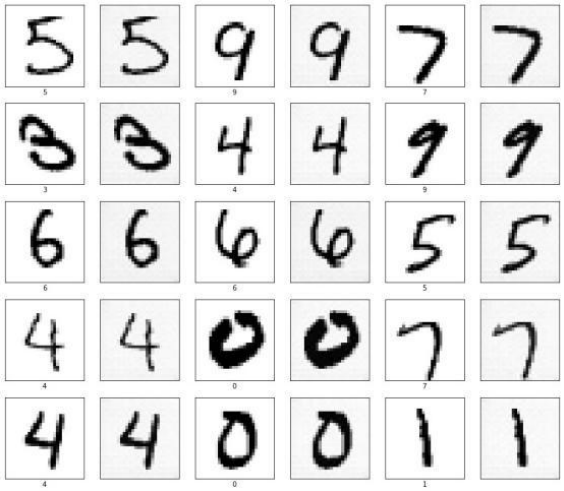

Fig. 2(d). Jump connection and negative feedback

Fig. 2. Generation Effect on Test Data

(The clear picture is input, the label of the picture is the result of classification and the blurry picture is the output of the network.)

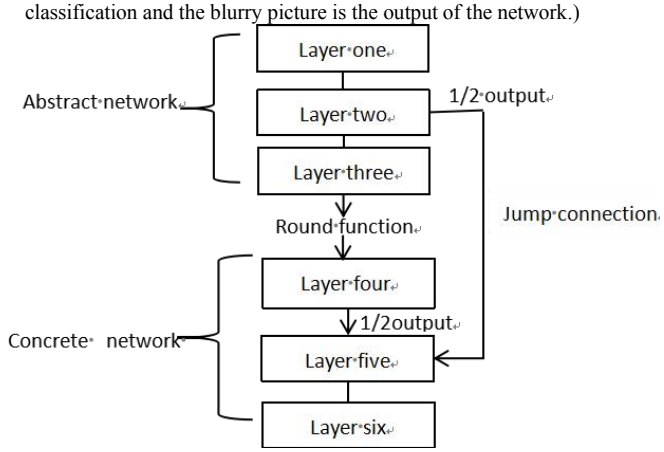

Fig. 3. Round function and jump connection

In order to generate the output which is very similar to input, we can use jump connection(which is in Fig3) and negative feedback. The output of one layer of abstract network which can be seen as the knowledge of features you have 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 five is $\mathrm{B}\pm\xi$, ξ is the error because the inverse function which used before is the approximation function. So $\frac{1}{2}(\mathrm{B}+\mathrm{B}\pm\xi)=B\pm\frac{1}{2}\xi$, the error decreases. The more jump connection(more knowledge about features), the less training time, the more similarity. It fits the process of learning. Fig. 2(c) shows the result of layer next to output layer of abstract network connecting with the layer next to input layer of concrete network. Fig. 2(d) shows the result of jump connection and negative feedback. We can see that the background is no longer dark when we use negative feedback. Why is feedback? Inspired by principle of automatic control, I add negative feedback. Because the whole network is like the proportional integral differential part of automatic control, our aim is to make output very similar to input.So let the the difference of input and output as the new input will decrease the difference of output and input. It is important to note that negative feedback only works when jump connection or regression .Because the error is small, when it propagate to softmax layer, the output of softmax layer will be close to 0, then it stop propagates.

The following is my thinking about this network and mind and consciousness of mankind. The abstract network is like abstract thought and the concrete network is like concrete thought. Meanwhile the concrete network is like memory. It generates input image in the brain, its' output can be seen as consciousness(the look of input in the brain). Man can remember only through the generation of concrete network. For example, I had been somewhere several years ago, when I am there once, I realize that I went there before. Because the old scene was generated in my brain, the concrete network has the parameter of it. When the new scene is input into brain, it generates old scene through the concrete network by the parameters, then takes the intersection of the old scene and the new scene. We went there before because of the result of intersection is large. Memory is always Common sense(The information remembered can be seen as common sense to individual). Lethe is that when new knowledge input, the training process makes the parameters change. The more new knowledge, the more changing, the more forgetting.

When the label has the multi-attribute[4], we can use this network to generate different things. My another paper is about multi-attribute label.

CONCLUSION

1.The network achieves the intended function through inverse function. The abstract network can classify and regress, the concrete network can generate input from concept or label.

2.We can change the input to any form by encoder and then change it back by decoder through inverse function. The concrete network can be seen as the memory stored by the parameters. Lethe is that when new knowledge input, the training process makes the parameters change.

REFERENCES

[1] Ian Goodfellow, Yoshua Bengio, Aaron Courville, "Deep Learning", MIT Press, 2016,pp 67-69

[2] Andrew L, Maas, Awni Y, Hannum and Andrew Y. Ng, "Rectified Nonlinearities Improve Neural Network Acoustic Models", ICML, 2013.

[3] Tensorflow Tutorials and Apis, google.inc.

[4] Jinxin Wei, Qunying Ren, "Multi-attribute Recognition,the Key to Universal Neural Network". unpublished