Explore the variants of the ReLU activation function in multi-layer networks trained on MNIST dataset

Man Wu

Abstract

In this small project, I investigate the performance of different activation functions in multi-layer networks to address the MNIST digit classification problem. I compare the performances of the five activation functions (Sigmoid, ReLU, Leaky ReLU, ELU and SELU) on the same database. I find that ReLU and its further functions have better performance than Sigmoid function and the variants of the ReLU activation function are better than ReLU. Then I choose one activation function to explore the impact of the depth of the network. After comparing the results of different numbers of hidden layers, I can see that the network with five hidden layers has the best performance in this specific case. In addition, I compare the different weight initialization strategies and in the experiment the Uniform distribution performs better than Gaussian distribution.

1. Introduction

In recent years, deep learning in the field of computer vision has made remarkable achievements, one of the important factors is the development of activation functions. As I know, there are many papers focusing on exploring the best activation function. I think it is meaningful to have some conclusions by myself from some experiments. Thus I need to answer the following questions from my experiments. The first one is the performance of different activation functions. Secondly, I need to explore the impact of the depth of the hidden layer to the network. Finally, I need to investigate different approaches to weight initialization. In all my experiments, the batch size is 50 and train for 100 epochs.

2. Activation functions

In this section, I mainly introduce the four activation functions: Rectified Linear Unit (ReLU)(Nair & Hinton, 2010), Leaky Rectified Linear Unit (Leaky ReLU)(Maas, Hannun, & Ng, 2013), Exponential Linear Unit (ELU)(Clevert, Unterthiner, & Hochreiter, 2015) and Scaled Exponential Linear Unit (SELU)(Klambauer, Unterthiner, Mayr, & Hochreiter, 2017). In the following subsections, I use equations to introduce each activation function mathematically.

2.1. Rectified Linear Unit

Formally, the ReLU is defined as:

$$\text{relu}(x) = \max(0, x), \tag{1}$$

which has the gradient:

$$\frac{d}{dx}\text{relu}(x) = \begin{cases} 0 & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{2}$$

2.2. Leaky Rectified Linear Unit

The Leaky ReLU has the following form:

$$\text{lrelu}(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{3}$$

which has the gradient:

$$\frac{d}{dx}\text{lrelu}(x) = \begin{cases} \alpha & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{4}$$

typically $\alpha = 0.01$ and in this experiment α= 0.01

2.3. Exponential Linear Unit

The equation of the ELU is:

$$\text{elu}(x) = \begin{cases} \alpha(exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{5}$$

which has the gradient:

$$\frac{d}{dx}\text{elu}(x) = \begin{cases} \alpha exp(x) & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{6}$$

typically $\alpha = 1$ and in this experiment $\alpha = 1$

2.4. Scaled Exponential Linear Unit

Mathematically, the SELU is defined as:

$$\text{selu}(x) = \lambda \begin{cases} \alpha(exp(x) - 1) & \text{if } x \leq 0 \\ x & \text{if } x > 0. \end{cases} \tag{7}$$

which has the gradient:

$$\frac{d}{dx}\text{selu}(x) = \lambda \begin{cases} \alpha exp(x) & \text{if } x \leq 0 \\ 1 & \text{if } x > 0. \end{cases} \tag{8}$$

in this experiment, $\alpha = 1.6733$ and $\lambda = 1.0507$

3. Experimental comparison of activation functions

In this section, I use five activation functions (Sigmoid, ReLU, Leaky ReLU, ELU and SELU) to carry out the experiments on the MNIST task. I use two hidden layers with 100 hidden units per layer for each experiment. The results are showed in Figure 1. From Figure 1, firstly we can see that ReLU and its three variants have faster convergence speed than Sigmoid function both on the training dataset and validation dataset. I guess the reason of the poor performance of Sigmoid function may be that it often causes the derivative to gradually change to 0 in the neural network. So the parameters can not be updated and the neural network can not be optimized. I think it is a good choice to use ReLU and its further functions as the activation function. Secondly, comparing the results of the ReLU family (ReLU, Leaky ReLU, ELU and SELU), the differences between these functions is very small, but we can see that the convergence of the three further functions related to ReLU is slightly faster than the ReLU function. Meanwhile, from Table 1, we can see the errors of Leaky ReLU, ELU and SELU are a little lower than ReLU. Maybe I can assume that the Leaky ReLU, ELU and SELU functions are better choices when selecting the activation function.

4. Deep neural network experiments

In this section, I use the Leaky ReLU function as the activation function to explore the impact of the depth of the network for MNIST. I compared the different performances of the Leaky ReLU function in different numbers of hidden layers from 2 to 8. As showed in Figure 2, the performances of the errors and accuracies in the training dataset nearly the same, so it is meaningless to compare the data in the training dataset. But we can see some features in the validation dataset. In general, as the number of hidden layer increases, the accuracy rate increases and the error rate decreases in the validation dataset. However, from Table 2, we can see that after the fifth layer the accuracies are becoming lower and the errors are becoming higher. From the results of the experiment, I can assume that maybe five hidden layers is the best choice for this experiment and more than five hidden layers can cause overfitting

About the part of weight initialization, I compare five initialization approaches, which are Uniform distribution (Fan-in, Fan-out , Fan-in and Fan-out), Gaussian distribution and SELU layers recommended by Klambauer et al. (2017). From Figure 3 and Table 3, we can see the performances of these different weight initialization strategies and the results has some features. Firstly, in the Uniform distribution, the Fan-in and Fan-out performs best and it has the highest accuracy rate and lowest error rate it the validation dataset. Secondly, the Uniform distribution performs better than the Gaussian distribution (Normal distribution). Thirdly, the SELU layers recommended by Klambauer et al. (2017) has the poorest performance (I don't know why, maybe there's something wrong with my experiment code.)

5. Conclusions

Finally, I have made some conclusions through all the experiments. ReLU and the three further functions related to ReLU have faster convergence speed than Sigmoid function. Meanwhile the Leaky ReLU, ELU and SELU functions perform better than ReLU function in the experiment. According to the question of the depth of hidden layers, I can assume that the numbers of hidden layer are not the higher the better and it depends on many factors. In this experiment, five hidden layers is a better choice than others. When comes into the weight initialization strategies, the Uniform distribution performs better than the Gaussian distribution (Normal distribution), particularly in the uniform distribution, the Fan-in and Fan-out method performs best. The rapid development of deep learning leads to the appearance of different forms of activation function. However, how to choose an activation function and which one is better becomes a facing problem and still need to rely on experimental guidance.

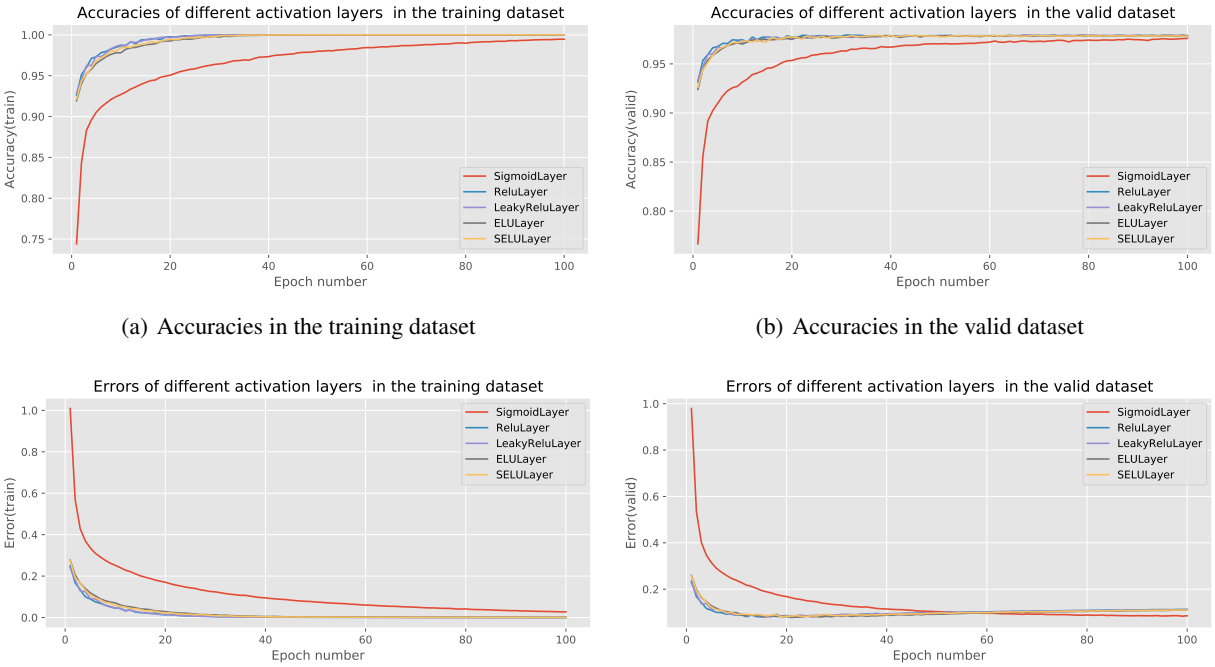
References

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*.

Klambauer, G., Unterthiner, T., Mayr, A., & Hochreiter, S. (2017). Self-normalizing neural networks. In *Advances in neural information processing systems* (pp. 972–981).

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml* (Vol. 30, p. 3).

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (icml-10)* (pp. 807–814).



(a) Accuracies in the training dataset

(b) Accuracies in the valid dataset

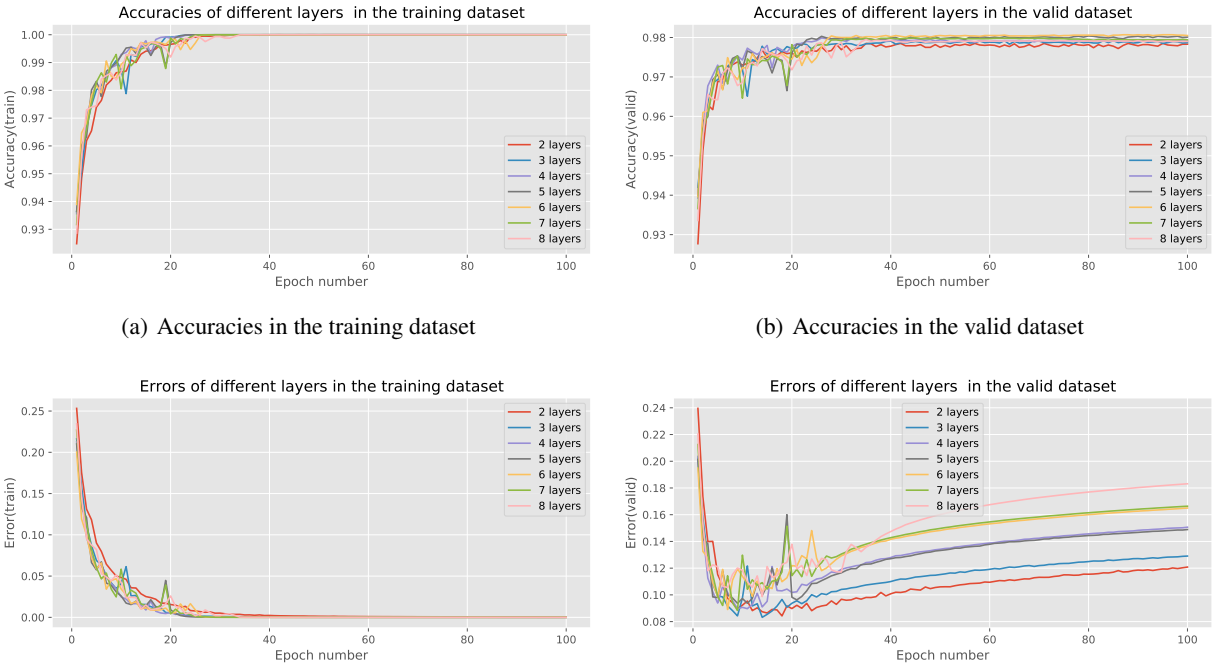
(c) Errors in the training dataset

(d) Errors in the valid dataset

Figure 1. The accuracies and errors of different activation functions in the training and valid dataset. Please use high resolution for better view.

Table 1. Classification accuracies and errors for different activation functions in the 100 epoches on train and valid data sets.

Data set	Sigmoid		ReLU		LeakyRelu		ELU		SELU	
	Acc(%)	Err(%)	Acc(%)	Err(%)	Acc(%)	Err(%)	Acc(%)	Err(%)	Acc(%)	Err(%)
Train dataset	99.5	2.75	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Valid dataset	97.6	8.54	97.9	11.3	97.9	11.2	97.8	11.0	97.8	11.0



(a) Accuracies in the training dataset

(b) Accuracies in the valid dataset

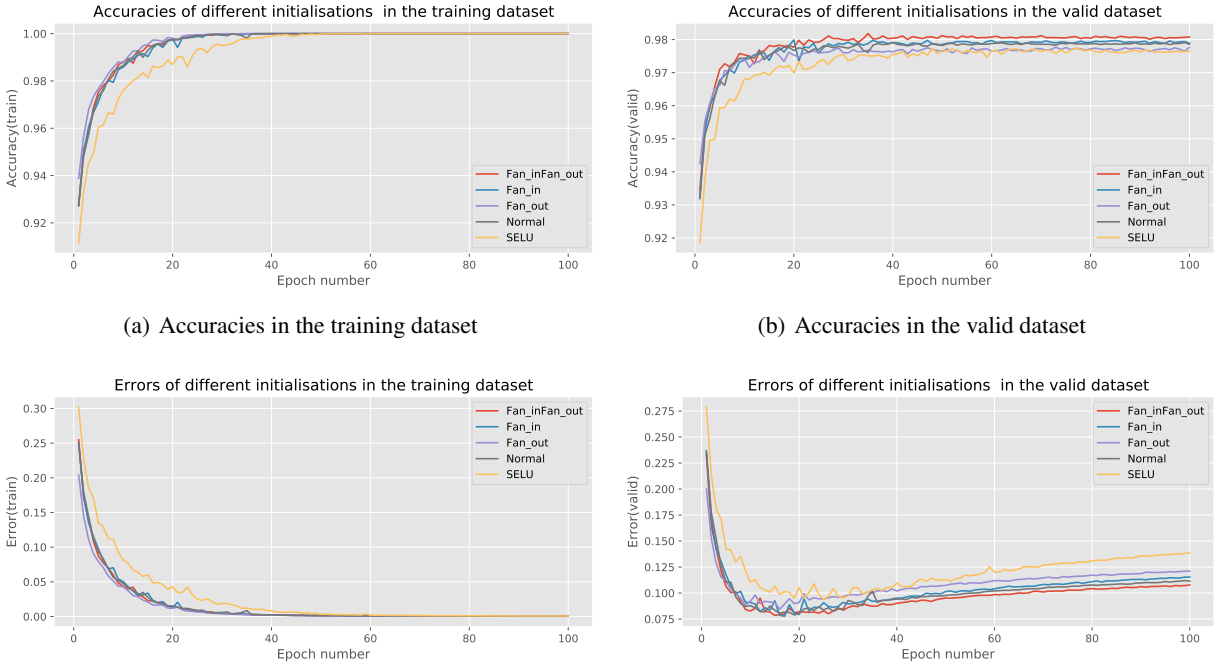
(c) Errors in the training dataset

(d) Errors in the valid dataset

Figure 2. The accuracies and errors of different layers in the training and valid dataset. Please use high resolution for better view.

Table 2. Classification accuracies and errors for different layers in the 100 epoches on train and valid data sets.

Data set	2 layers		3 layers		4 layers		5 layers		6 layers		7 layers		8 layers	
	Acc	Err	Acc	Err	Acc	Err	Acc	Err	Acc	Err	Acc	Err	Acc	Err
Train dataset	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Valid dataset	97.8	12.1	97.9	12.9	97.9	15.2	98.0	14.9	98.0	16.5	97.9	16.6	97.9	18.3



(a) Accuracies in the training dataset

(b) Accuracies in the valid dataset

(c) Errors in the training dataset

(d) Errors in the valid dataset

Figure 3. The accuracies and errors of different different weight initialisation strategies in the training and valid dataset. Please use high resolution for better view.

Table 3. Classification accuracies and errors for different weight initialisation strategies in the 100 epoches on train and valid data sets.

Data set	Fan_inFan_out		Fan_in		Fan_out		Normal		SELU	
	Acc	Err	Acc	Err	Acc	Err	Acc	Err	Acc	Err
Train dataset	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0	100.0	0.0
Valid dataset	98.1	10.8	97.9	11.5	97.8	12.1	97.9	11.2	97.6	13.9