International Journal of Advanced Smart Convergence Vol.6 No.4 73-79 (2017)
http://dx.doi.org/10.7236/IJASC.2017.6.4.11

IJASC 17-4-11

The Effect of Hyperparameter Choice on ReLU and SELU Activation Function

Pratama Kevin[†], Dae-Ki Kang[*]

[*]*Department of Ubiquitous IT, Graduate School, Dongseo University, Busan, Korea*
[†]*Department of Computer Engineering, Dongseo University, Busan, Korea*
[*]*kvpratama@gmail.com,* [†]*dkkang@dongseo.ac.kr*

Abstract

The Convolutional Neural Network (CNN) has shown an excellent performance in computer vision task. Applications of CNN include image classification, object detection in images, autonomous driving, etc. This paper will evaluate the performance of CNN model with ReLU and SELU as activation function. The evaluation will be performed on four different choices of hyperparameter which are initialization method, network configuration, optimization technique, and regularization. We did experiment on each choice of hyperparameter and show how it influences the network convergence and test accuracy. In this experiment, we also discover performance improvement when using SELU as activation function over ReLU.

Keywords: *Deep Learning, Convolutional Neural Network, Hyperparameter, Activation Function, ReLU, SELU*

1. Introduction

Convolutional Neural Network (CNN) is one of Neural Networks model that has proven to produce an excellent result in areas such as image recognition and classification. CNN is very similar to the normal Fully Connected Neural Networks. The main difference between the two networks is the convolution process in CNN. The main purpose of this convolution process is to extract features from input image. Convolution process will preserve the spatial relationship between pixels by learning features of an image by sliding small squares of matrix over the entire image. This small matrix is known as filter or kernel. With this change of architecture, CNN has been very successful in image related tasks such as identifying faces, classifying images, detect objects in images and much more application.

Optimizing deep learning algorithm involve choosing hyperparameter of the neural network before training the model. Neural network can have many hyperparameters and it could be a tedious work to select the right hyperparameter for the model. One of hyperparameter in deep neural network is the choice of activation function. Activation function is a function to perform a transformation on the input it receives. The purpose is to keep the value that has been multiplied by weight and added by bias, within a manageable range.

This paper will evaluate the performances of two activation function, the currently most popular, ReLU function, and its modification, SELU that is newly proposed in 2017. These two activation function will be

Manuscript Received: Nov. 10, 2017 / Revised: Nov. 25, 2017 / Accepted: Dec.7, 2017
Corresponding Author: dkkang@dongseo.ac.kr
Tel: +82-51-320-1724, Fax: +82-51-327-8955
Department of Computer Engineering, Dongseo University, Korea

analyzed on CNN model on four different choices of hyperparameter. The chosen hyperparameter are the most commonly tuned to improve the performance of the neural network. The four hyperparameters are initialization method, network configuration, optimization technique, and regularization. We then show the result of our experiment and draw conclusion from it.

2. Activation function

The Rectified Linear Unit or ReLU is currently the most popular activation function in deep learning era. It was found to speed up the convergence of neural network when compared to sigmoid or tanh functions. It is believed that this advantage is because of its non-saturating form. The other advantage of ReLU over its saturated counterpart is the simplicity of operation. ReLU can be implemented by thresholding an input of activation at zero. The ReLu is given by the equation below:

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

The Scaled Exponential linear units or SELU [3] is one of modification of ReLU function. The main idea of this activation function is to keep the network activation in a certain mean and variance value. The mean and variance can be any values but the original paper suggest mean value as 0 and variance as 1. The SELU equation is given as:

$$f(x) = \begin{cases} \lambda x, & \text{if } x > 0 \\ \lambda\alpha e^{x} - \lambda, & \text{if } x \leqslant 0 \end{cases} \tag{2}$$

To achieve the mean and variance value as suggested by the paper the scale factor of λ is set to 1.6733 and the scale factor of α is set to 1.0507. The author of SELU has proved that the output of this function will stay within the mean and variance value even in a very deep neural network.

3. Experiments

In machine learning and deep learning, we can handle various types of datasets such as image, sound, and text. These datasets can have multiple dimensions or scales. A feature that has smaller scale might play a tiny role compare to the feature that has a bigger scale. However, both feature may contain important information that will be useful for the task we want to perform. To deal with this situation, we will normalize the features of the dataset independently to the same scale, so they contribute equally while performing the task. The method we use to do normalization is Z-score normalization. The formula is given below,

$$z = \frac{x-\mu}{\sigma} \tag{3}$$

with μ as the distribution mean and σ as the standard deviation. This method is widely used for normalization in many machine learning algorithms (e.g., support vector machines, logistic regression, and neural networks). The result of normalization is the features will be rescaled to have the properties of a standard normal distribution with zero-mean and unit-variance. Standardizing is not only important if we are comparing measurements that have different units, but it is also a general requirement for many machine learning algorithms.

3.1 Initialization method

The first hyperparameter we will experiment with is initialization method. When training deep neural network, initializing the weights of the network can be the determining factor of successful convergence.

Initialize the right weight can also lead to a faster convergence. Initialize weight with small value will make the variance of the input signal shrink as it passes through layer in the network. Eventually, the input will drops to a very small value it become useless for learning. Similarly, if the weight is initialized with a large value the variance of input data tends to increase rapidly with every passing layer and become too massive to be useful.

Initializing the network with the right weights (not too small or too big) is very important. Unfortunately, it seems to be a random process as we don't know the weight that will work for our data. A group of researcher [1] proposes a new method of initialization that makes sure the weights are initialized properly and have a reasonable range. The main idea of Xavier initialization is to maintain the variance of the weight to remain the same as it passes through layers. This helps the network to keep the input signal from exploding to a high value or vanishing to zero.

$$Var(w_i) = \frac{2}{n_{in}+n_{out}} \tag{4}$$

Xavier initialization is derived based on the assumption that the neural network is using symmetric activation function. It also assumed that the weights are initialized independently and both input and weights have zero mean.

Building on Xavier initialization, He et al.[2] come up with different initialization method. Contrary to Xavier initialization, He initialization derive based on a non-linear activation function and assume that both input and weight do not have zero mean. This lead to a different conclusion to Xavier initialization.

$$Var(w_i) = \frac{2}{n_{in}} \tag{5}$$

This paper evaluates the effect of Xavier and He initialization on Convolution Neural network with ReLU or SELU as activation function. The result can be seen in Table 1 and Table 2. Additionally, this paper also evaluates the initialization method that is used in Self-Normalizing Neural Networks (SNN) paper where the variance calculation of weight is given in the formula below.

$$Var(w_i) = \frac{1}{n_{in}} \tag{6}$$

From the tables 1 and 2, we can see the initialization method from SNN paper works best for SELU network. According to the author, the initialization for SNN will help the output of SNN model to stay at normalized point at zero mean and unit variance which could be the reason it is best suited for SeLU network. The interesting point is that SNN initialization also works for ReLU network with test accuracy outperform SELU network on SVHN dataset.

Table 1. Test accuracy on Cifar-10 for Xavier and He

	Xavier	He	SNN Init
ReLU	73.61%	73.97%	75.13%
SELU	74.64%	73.98%	75.77%

Table 2. Test accuracy on SVHN for Xavier and He

	Xavier	He	SNN Init
ReLU	91.56%	91.33%	92.65%
SELU	90.68%	91.80%	91.85%

3.2 Network configuration

This paper compare four different network configuration. The network configuration is given in Table 3

From Table 4 reader can observe that SELU network gain a better test accuracy as the network goes deeper and bigger on both datasets. This behavior is not seen in ReLU network. A deeper ReLU network doesn't get a significant improvement in term of accuracy with accuracy goes down in model 2 and 3 on Cifar-10 dataset. Similar behavior can be observed in Table 5 where deeper SELU network gets a bigger improvement in term of test accuracy compares to ReLU network.

Table 3. Network configuration

	Model I	Model II	Model III	Model IV
Conv	64, 5x5, 2	64, 5x5, 2	64, 5x5, 2	64, 5x5, 2
			64, 5x5, 2	64, 5x5, 2
			64, 5x5, 2	64, 5x5, 2
Pool	3x3, 2	3x3, 2	64, 5x5, 2	64, 5x5, 2
			3x3, 2	3x3, 2
Conv	64, 5x5, 2	64, 5x5, 2	64, 5x5, 2	128, 5x5, 2
		64, 5x5, 2	64, 5x5, 2	128, 5x5, 2
		64, 5x5, 2	64, 5x5, 2	128, 5x5, 2
		64, 5x5, 2	64, 5x5, 2	128, 5x5, 2
		64, 5x5, 2	64, 5x5, 2	128, 5x5, 2
Pool	3x3, 2	3x3, 2	64, 5x5, 2	128, 5x5, 2
			3x3, 2	3x3, 2
Conv	128, 3x3, 2	128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
	128, 3x3, 2	128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
	128, 3x3, 2	128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
		128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
		128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
		128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
		128, 3x3, 2	128, 3x3, 2	256, 3x3, 2
Pool	3x3, 2	3x3, 2	3x3, 2	3x3, 2

3.3 Optimization techniques

This section will compare two Gradient Optimization Techniques. The first technique is Momentum [4]. Momentum is a method to help accelerate gradient descent to reach convergence. It does that by navigating gradient descent along the relevant direction and dampens oscillations. Momentum adds a fraction γ from the update from the past step to the current update.

$$v_t = \gamma v_{t-1} + \eta \nabla J(\theta) \tag{7}$$

$$\theta = \theta - v_t \tag{8}$$

Table 4. Test accuracy on Cifar-10 for different configurations

	Model I	Model II	Model III	Model IV
Conv	73.97%	69.88%	68.70%	70.12%
Pool	75.77%	76.23%	77.28%	80.19%

Table 5. Test accuracy on SVHN for different configurations

	Model I	Model II	Model III	Model IV
Conv	91.33%	92.51%	92.99%	93.47%
Pool	91.85%	92.77%	93.46%	94.17%

The second optimization technique is Adam [5]. Adam stands for Adaptive Moment Estimation. It is a method that computes adaptive learning rate for each parameter. Adaptive learning rate means it adapts the learning rate to the parameters, performing larger updates for infrequent and smaller updates for frequent parameters.

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t}+\varepsilon} m_t \tag{9}$$

We set the momentum term γ with 0.9. For Adam optimization, this paper will use the default value proposed by the Adam author which is 0.9 for β_1, 0.999 for β_2, and 10^{-8} for ε. The result can be seen in Table 6 and 7.

Table 6. Test accuracy on Cifar-10 for Momentum and Adam

	Momentum	Adam
ReLU	53.61%	73%
SELU	52.16%	75.89%

Table 7. Test accuracy on SVHN for Momentum and Adam

	Momentum	Adam
ReLU	79.75%	91.12%
SELU	82.27%	91.91%

It is obvious that Adam optimizer produces a better result compare to the Momentum optimizer. The author of Adam describes this method as combining the advantages of two other optimization method, AdaGrad and RMSProp. Not only adapting the parameter learning rate on the average first moment as in RMSProp, Adam also uses the average of the second moments of the gradient. This combination has led to a notable improvement from Momentum optimizer.

3.4 Regularization

This paper will observe the impact of two regularization method on CNN model. The first regularization method is L2 regularization. L2 regularization adds a penalty equal to the sum of the squared of value of the weights to the error term. L2 regularization will force the parameters to be small since the bigger the L2

regularization penalty, the smaller the weight.

The second regularization method is dropout [6]. Dropout randomly ignores neurons during training which means the neurons contribution is temporally removed. When neurons are randomly ignored during training, the other neurons will step in to handle the representation required to make a prediction. This makes the network became less dependent on the specific neurons, resulting in a network with better generalization and less likely to overfit the training data.

Table 8. Test accuracy on Cifar-10 for L2 and Dropout

	Momentum	Adam
ReLU	73.48%	75.78%
SELU	75.95%	76.63%

Table 9. Test accuracy on SVHN for L2 and Dropout

	Momentum	Adam
ReLU	91.68%	93.04%
SELU	91.43%	92.16%

From the tables 8 and 9, the reader will notice a superior performance of dropout compares to L2 regularization. Randomly shut-off neurons during training prevent co-adaptation among them, therefore forcing them to be less dependent on other neurons to correct its mistake. Dropout allow a single network to model a large number of sub-networks in inexpensive way, making it a more robust regularization than L2 regularization.

4. Conclusion

This paper has compared ReLU and SELU network with four hyperparameters (initialization method, network configuration, optimization technique, and regularization). All four hyperparameter give various effect (negatively and positively) on both network in term of test accuracy and speed of convergence.

One clear advantage of SELU network over ReLU network that is observed from this paper is its speed of convergence regardless the choice of hyperparameter. The only time that ReLU network beat SELU network is when Momentum is selected as optimization technique.

One more advantage of SELU network is its ability to use a big and deep network and gain a better test accuracy. This is clearly observed as the test accuracy become better with a deeper network with the deepest SELU network in this paper gain the best test accuracy. Meanwhile, there is no obvious sign of this behavior with ReLU network.

Acknowledgement

This research was supported by 2016 Dongseo University research grants and the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (no. NRF-2015R1D1A1A01061328).

References

[1] X. Glorot and Y. Bengio, "Understanding the Difficulty of Training Deep Feedforward Neural Networks", *In Proc. AISTATS. Society for Artificial Intelligence and Statistics,* 2010.

[2] K. He, X. Zhang, S. Ren and J, Sun, "Delving Deep into Rectifiers: Surpassing Human-level Performance on Imagenet Classification", *In Proceedings of the IEEE International Conference on Computer Vision,* 2015.

[3] G. Klambauer, T. Unterthiner, A. Mayr and S. Hochreiter, "Self-Normalizing Neural Networks", *CoRR,* 2017, abs/1706.02515.

[4] N. Qian, "On the Momentum Term in Gradient Descent Learning Algorithms", *Neural Networks : The Official Journal of The International Neural Network Society,* Vol. 12, No. 1, pp. 145–151, 1999.

[5] D. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization", *International Conference on Learning Representations,* pp. 1–13, 2015.

[6] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving Neural Networks by Preventing Coadaptation of Feature Detectors," 2012, http://arxiv.org/abs/1207.0580.