Journal of Electrical Engineering 7 (2019) 1-7
doi: 10.17265/2328-2223/2019.01.001

Training of Multi-layered Neural Network for Data Enlargement Processing Using an Activity Function

Betere Job Isaac[1], Hiroshi Kinjo[2], Kunihiko Nakazono[2] and Naoki Oshiro[2]

1. Mechanical Systems Engineering Course, Graduate School of Engineering and Science, University of the Ryukyus Senbaru 1, Nishihara, Okinawa 903-0213, Japan

2. Faculty of Engineering, University of the Ryukyus Senbaru 1, Nishihara, Okinawa 903-0213, Japan

Abstract: In this paper, we present a study on activity functions for an MLNN (multi-layered neural network) and propose a suitable activity function for data enlargement processing. We have carefully studied the training performance of Sigmoid, ReLu, Leaky-ReLu and L & exp. activity functions for few inputs to multiple output training patterns. Our MLNNs model has L hidden layers with two or three inputs to four or six outputs data variations by BP (backpropagation) NN (neural network) training. We focused on the multi teacher training signals to investigate and evaluate the training performance in MLNNs to select the best and good activity function for data enlargement and hence could be applicable for image and signal processing (synaptic divergence) along with the proposed methods with convolution networks. We specifically used four activity functions from which we found out that L & exp. activity function can suite DENN (data enlargement neural network) training since it could give the highest percentage training abilities compared to the other activity functions of Sigmoid, ReLu and Leaky-ReLu during simulation and training of data in the network. And finally, we recommend L & exp. function to be good for MLNNs and may be applicable for signal processing of data and information enlargement because of its performance training characteristics with multiple teacher training patterns using original generated data and hence can be tried with CNN (convolution neural networks) of image processing.

Key words: Data enlargement processing, MLNN, activity function, multi teacher training signals, BP NN, CNN.

1. Introduction

In recent years, the application of NN (neural networks) about intelligent systems has attracted attention and lots of papers and applications have been presented. To mention a few, image processing and recognition technology in intelligent systems occupy a big percentage. It has been put to practical use in scenes where large amounts of data are handled and have been proved by many increasing numbers of researchers [1-3] especially for signal processing as to manage data transfer effectively with other systems to overcome conventional means particularly [4].

Many scientists have undergone a resurgence in computation research community from the field of NNs and deep machine learning. Furthermore, recently more research has been reported on multi hidden layer NN training [5]. Earlier scientists were motivated in large part by visions of imbuing computer programs with life-like ability to self-replicate and with adaptive capability to learn the environment in terms of artificial intelligence. The application of NNs about image processing and recognition technology in intelligent control systems indicates poor training performance of the sigmoid function due to the gradient disappearance problem with BP (backpropagation) NN training [6, 7]. It is also said that ReLu function is good compared to other activity functions in the field of 2D image processing and recognition [8]. We have also noted that Leaky-ReLu function is almost like ReLu function although with a slight improvement on the training results and can accommodate more characteristic values for BP NN training. Therefore, our motivation interest was to further investigate more by bringing

Corresponding author: Betere Job Isaac, MSc, research fields: robotics, artificial intelligent control systems and signal processing.

more activity functions into play to select one suitable for DENN (data enlargement neural network).

In this study, we extend our proposed L & exp. activity function for multi logic training patterns with multi-layered neural training networks [9, 10]. We consider it to be good for data enlargement and can truly overcome the drawbacks of sigmoid, ReLu and Leaky-ReLu functions as used in this study using original generated data sets. We hope it can be adopted and applied to CNN (convolution neural networks) in 2D image and signal processing to examine the training performance with MLNN (multi-layered neural networks) for artificial intelligent systems.

2. NN Model

NNs are typically organized in layers. Fig. 1 shows the NN model we used in this study. Layers are made up of several interconnected nodes which contain an activation function. Patterns are presented to the network via the input layer which communicates to 5 neurons and 5 hidden layers where the actual processing is done via a system of weighted connections respectively.

The hidden layers then link to an output layer and gives the result of the desired output as shown in the feedforward NN in Fig. 1, where I, J & K are number of neurons of input layer, hidden layers and output layer respectively with L hidden layers, and t_1, t_2, t_3, t_4, t_5 and t_6 as teacher training signals.

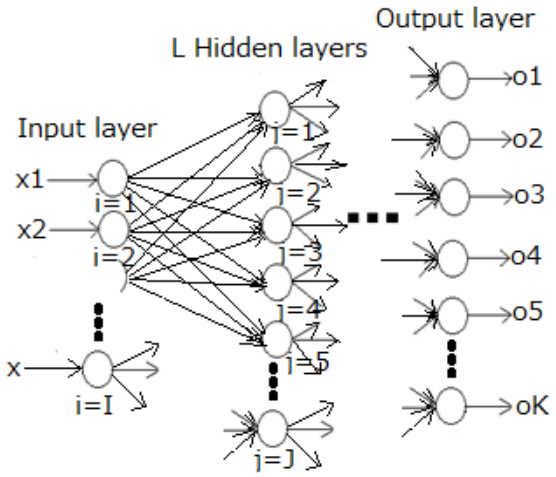


Fig. 1 MLNN.

The input/output relation of the NN is given by the following equations.

$$o_i = f^I(x_i), i = 1, 2, \dots, I \quad (1)$$

$$o_j{}^{(l)} = f^H\left(net_j{}^{(l)}\right), l = 1,2, \dots, L, j = 1, 2, \dots, J \quad (2)$$

$$net_j{}^{(l)} = \begin{cases} \sum_i w_{ji} o_i, l = 1 \\ \sum_j w_{jj}{}^{(l)} o_j{}^{(l-1)}, l \geq 2 \end{cases}$$

$$o_k = f^O(net_k), k = 1, 2, \dots, K \quad (3)$$

$$net_k = \sum_j w_{kj}\, o_j{}^{(L)}$$

where, o_i and o_j are outputs of input and hidden layers respectively, w_{ji} w_{jj} and w_{kj} are connecting weights. $f^I(.), f^H(.)$ and $f^O(.)$ are activity functions of input, hidden and output layers respectively. Many methods of NNs for signal processing have been well studied and applied for many industrial problems. There are many network types consisting of many inputs with few outputs and it is useful for image processing (CNN).

However, we consider the other type of the network construction as in Fig. 1 in respect to our motivation where the network could have multiple outputs from few inputs. This network is taken to be applicable to data enlargement fields. Simulations were basically concentrated on the functions as follows.

Sigmoid function:

$$f^H(x) = \frac{1}{1+exp^{(-x)}} \quad (4)$$

ReLu function:

$$f^H(x) = \begin{cases} x\ 0 \leq x \\ 0\ 0 > x \end{cases} \quad (5)$$

Leaky-ReLu function:

$$f^H(x) = \begin{cases} x\ 0 \leq x \\ \alpha x\ 0 > x \end{cases} \quad (6)$$

where, α is the slope gradient of the function.

And the L & exp function as

$$f(x) = \begin{cases} x + \beta\ x \geq 0 \\ \beta\, e^x\ x < 0 \end{cases} \quad (7)$$

where, β is the intercept. This function combined both

linear part and exponential part, so we called it L & exp. function.

In this study, we have used the input and output activity function as follows:

$$f^I(x) = f^O(x) = x \tag{8}$$

3. BP for MLNN

BP training is a gradient descent algorithm. It tries to improve the performance of the neural net by reducing its error along its gradient. The error is expressed by the RMSE (root-mean-square error), which can be calculated by the error function E for BP as shown in the following equation.

$$\boldsymbol{E} = \frac{\mathbf{1}}{\mathbf{2}} \sum_{\boldsymbol{p=1}}^{\boldsymbol{p}} \sum_{\boldsymbol{k=1}}^{\boldsymbol{k}} \left(t_k{}^{(p)} - o_k\right)^2 \tag{9}$$

where, the error E is half the sum of the geometric averages of the difference between the desired output $t_k^{(p)}$ and the actual output o_k over all patterns p. In each training step, the weights w_{ji}, w_{jj} and w_{kj} are adjusted towards the direction of maximum decrease and scaled by some learning rate epsilon ε as shown in the following modified equation of the synaptic connection weight vector W.

$$W^{(new)} = W^{(old)} - \varepsilon \frac{\partial E}{\partial W} \tag{10}$$

The generalized delta rule of BP is applied, and the gradient is as follows with the output layer:

$$\frac{\partial E}{\partial W} = -\delta_k \, o_j{}^{(l)} \tag{11}$$

where, δ_k is as follows:

$$\delta_k = (t_k - o_k) f^o{}'(net_k) \tag{12}$$

and, for the hidden layers as follows:

$$\frac{\partial E}{\partial W} = -\delta_j{}^{(l)} o_j{}^{(l-1)} \tag{13}$$

where, $\delta_j{}^{(l)}$ is as follows:

$$\delta_j{}^{(l)} = \sum \{\delta_j{}^{(l+1)} \, w_{jj}{}^{(l+1)}\} \, f^H{}'\left(net_j{}^{(l)}\right), l = L-1, L-2, \ldots, 2, 1 \tag{14}$$

where, $f^H{}'(.)$ is the derivative of the activity function.

Derived sigmoid function:

$$f^H{}'(x) = f^H(x)(1 - f^H(x)) \tag{15}$$

Derived ReLu function as:

$$f^H{}'(x) = \begin{cases} 1 & 0 \le x \\ 0 & 0 > x \end{cases} \tag{16}$$

Derived Leaky-ReLu function:

$$f^H{}'(x) = \begin{cases} 1 & 0 \le x \\ \alpha & 0 > x \end{cases} \tag{17}$$

And the derived L & exp. function as:

$$f^H{}'(x) = \begin{cases} 1 & x \ge 0 \\ \beta \, e^x & x < 0 \end{cases} \tag{18}$$

4. Experiment Simulations

Tables 1 and 2 show the parameters we used to build the NN training model for our study. We assumed these parameters to design a simple data enlargement training network from our original generated data sets to evaluate the training performance of various activity functions in an MLNN.

Tables 3 and 4 show the training patterns. We have two sets of training patterns representing two inputs to four outputs and three inputs to six outputs for training the data in the network. In here, x_1, x_2 and x_3 are input signals used to generate teacher training signals by magnifying each input signal twice or by three signals and more (replicated) depending on the volumes of data intended to train. t_1, t_2, t_3, t_4, t_5 and t_6 are the teacher training signals for both normal data and inverted data. Normal data are taken to be an expansion

Table 1 Constant parameter of the DENNs.

No.	Parameters	Value/method
1	No. of neurons in input layer (I)	2 or 3
2	No. of neurons in hidden layer (J)	12
3	No. of hidden neuron layers (L)	1-5
4	No. of neurons in output layer (K)	4 or 6
5	Activity functions	Sigmoid ReLu Leaky-ReLu ($\alpha = 0.2$) L & exp. ($\beta = 0.2$)

Table 2 Constant parameter of the BP.

No.	Parameters	Value/method
1	Training coefficient (ε)	0.1
2	Iterations	3,000

Table 3 Training pattern for four outputs.

Normal Data

No.	Inputs		Teacher signals			
	x_1	x_2	t_1	t_2	t_3	t_4
1	0	0	0	0	0	0
2	0	1	0	0	1	1
3	1	0	1	1	0	0
4	1	1	1	1	1	1

Inverted Data

No.	Inputs		Teacher signals			
	x_1	x_2	t_1	t_2	t_3	t_4
1	0	0	1	1	1	1
2	0	1	1	1	0	0
3	1	0	0	0	1	1
4	1	1	0	0	0	0

Table 4 Training patterns for six outputs.

Normal Data

No.	Inputs			Teacher signals					
	x_1	x_2	x_3	t_1	t_2	t_3	t_4	t_5	t_6
1	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1	0	0
4	1	0	1	1	1	0	0	1	1
5	1	1	0	1	1	1	1	0	0
6	1	1	1	1	1	1	1	1	1

Inverted Data

No.	Inputs			Teacher signals					
	x_1	x_2	x_3	t_1	t_2	t_3	t_4	t_5	t_6
1	0	0	0	1	1	1	1	1	1
2	0	0	1	1	1	1	1	0	0
3	0	1	0	1	1	0	0	1	1
4	1	0	1	0	0	1	1	0	0
5	1	1	0	0	0	0	0	1	1
6	1	1	1	0	0	0	0	0	0

of the original input signal twice to have an increment on data output to represent data enlargement. Inverted data are generated from the normal data by interchanging the teacher training signals and maintaining the inputs signals for both situations. We considered a simple case of DENN (data enlargement neural network) to generate the training data sets in Tables 3 and 4 basing on 0 and 1 signals as input and

Table 5 Successive rate for four outputs [%].

Normal Data

L	1	2	3	4	5
Sigmoid	100	100	2	0	0
ReLu	100	99	100	100	98
Leaky-ReLu	100	100	100	100	100
L & exp.	100	100	100	100	100

Inverted Data

L	1	2	3	4	5
Sigmoid	100	100	4	0	0
ReLu	0	0	0	0	0
Leaky-ReLu	0	0	0	0	0
L & exp.	100	100	100	100	99

Table 6 Successive rate for six outputs [%].

Normal Data

L	1	2	3	4	5
Sigmoid	100	100	7	0	0
ReLu	100	100	99	98	21
Leaky-ReLu	100	100	100	100	99
L & exp.	100	100	100	100	92

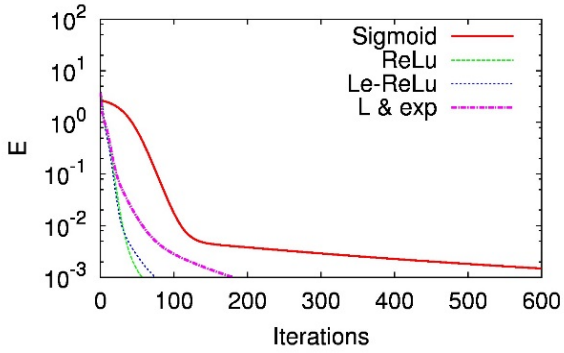
Inverted Data

L	1	2	3	4	5
Sigmoid	100	100	22	0	0
ReLu	0	0	0	0	0
Leaky-ReLu	0	0	0	0	0
L & exp.	89	100	100	96	81

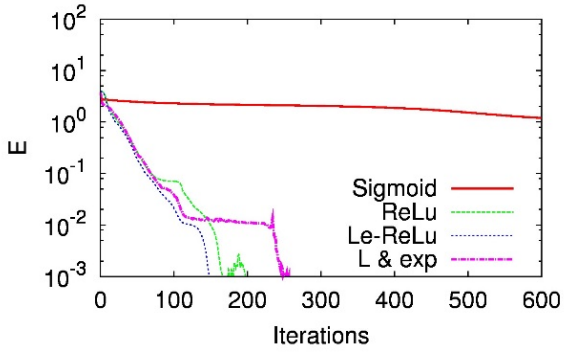
output respectively for our DENN model as shown by Fig. 1 in this study. It is because we believe and consider this to be the fundamental task to discuss the training performance of MLNN training without depending on data sets such that our results can easily be confirmed.

Tables 5 and 6 show the successive training rate percentages of each activity function training performance, where, *L* is the number of hidden neuron layers.

Figs. 2-5 show the training results for normal and inverted data, where, *E* on the vertical axis is the mean value error function against iterations on the horizontal axis. The successive training condition is when $E < 0.001$.

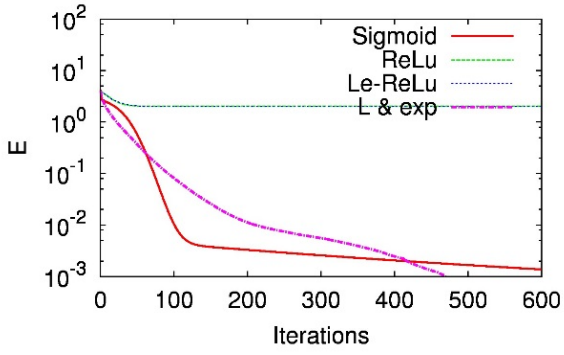


(a) $L = 1$

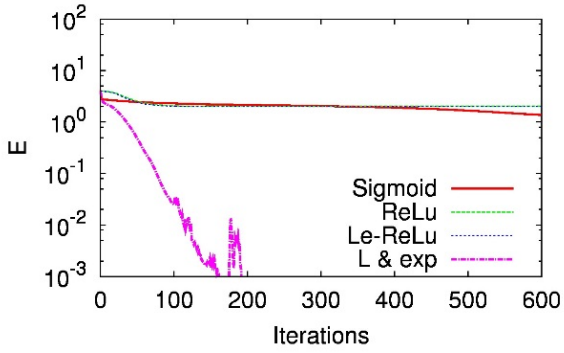


(b) $L = 3$

Fig. 2 Training results for four outputs normal data.

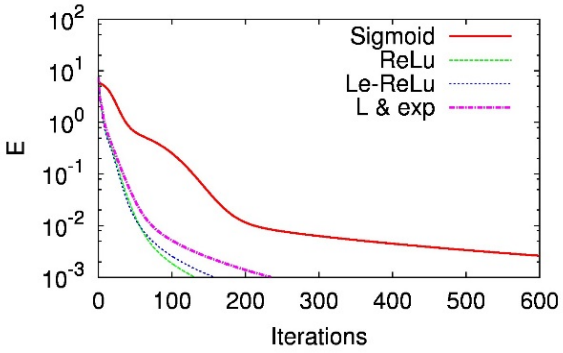


(a) $L = 1$

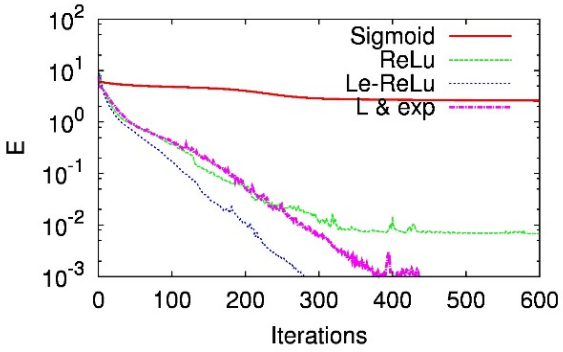


(b) $L = 3$

Fig. 3 Training results for four outputs inverted data.

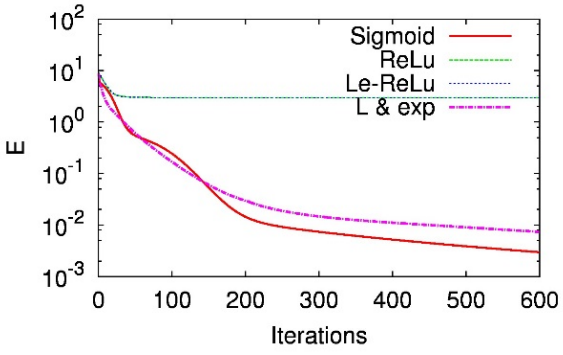


(a) $L = 1$

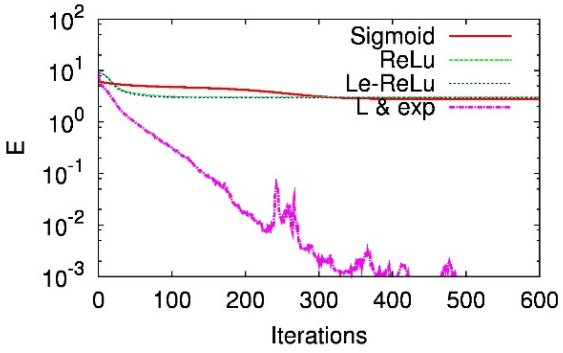


(b) $L = 3$

Fig. 4 Training results for six outputs normal data.



(a) $L = 1$



(b) $L = 3$

Fig. 5 Training results for six outputs inverted data.

5. Discussion

As per the results after training, it is seen in Tables 5 and 6 that L & exp. activity function trains all the patterns for both normal and inverted data for multi teacher training signals with the highest percentage rate in the network. Sigmoid could only train with few layers and it could appear to be good with only two layers in the network. ReLu function trained with only normal data output case and failed out when the data are inverted. Leaky-ReLu function is also seen to behave the same way since it appears to be an extended improvement of a ReLu function and seems to be good and fast during training of normal data like the ReLu function. But for wide data expression and deep training, we can see that Sigmoid, ReLu and Leaky-ReLu functions could not train satisfactorily well (Tables 5b and 6b) as L & exp. function especially when there is any change in the multi teacher training signals with many neurons and multiple layers in the network.

We considered the fact that L & exp. function is good and has an advantage over the other activity functions since it continued to satisfactorily train all patterns for both data situations without any limitation of the gradient values in the network. It can accommodate and train both positive and negative values with the increase in the number of layers and neurons in the network. It can therefore perform well with manipulated data in the any kind of system and give good results hence can be incorporated with more artificial complicated systems to handle many training parameters in any NN system and has got an advantage as compared to the other activity functions.

6. Conclusion

In this study, we investigated the training of an MLNN by an activity function for data enlargement processing. We have proved that BP training for an MLNN can give better training results using our original generated data sets as multi teacher training patterns by an activity function. In this article, few inputs to many outputs are considered to represent data enlargement where each input signal could train two teacher training signals in a four-output network. This process continued for various sets of data to confirm good training performance. We noted that L & exp. function trained all the patterns in all outputs without any interference and gave good percentage training results as compared to the other activity functions we used in the network. Thus, we propose it to be a suitable activity function to handle wide volumes of parameters in terms of data and other complicated artificial intelligence systems of CNN and machine deep learning systems. However, in this study, we have also seen that in the situation where data flows normally, ReLu and Leaky-ReLu functions are able and good to train all the patterns and very fast during training simulations, hence, can be applied to moderate data application systems. Sigmoid activity function degrades and fades out with BP especially when the numbers of layers and neurons are increased as seen for the 3rd neurons hidden layer training in the network.

For future work, we are focusing on more applications of this proposed function using the existing data sets especially in 2D image CNN and examine its training performance for image recognition and processing. We also need to find out why other activity functions like ReLu and Leaky-ReLu failed to train with MLNNs especially for data inverse training situations. Lastly, we want to examine the training performance of CNNs and MLNN in artificial intelligent systems.

References

[1] Rumelhart, D. E., McClelland, J. L., and the PDP Research group. 1986. *Parallel Distribution Processing.* MIT Press.

[2] Hassoun, M. H. 1995. *Fundamentals of Artificial Neural Networks.* MIT Press.

[3] Anderson, J. A., and Rosenfeld, R. 1988. *Neuro Computing Foundations of Research.* MIT Press.

[4] Lewis, F. L., Jagannathan, S., and Yesidirek, A. 1998. *Neural Network Control of Robot Manipulators and Nonlinear Systems*. Taylor & Francis.

[5] Kodaka, T., and Murakami, K. 2016. *Machine Learning*

and Deep Learning, Simulation by C programing, Ohmsha. (in Japanese)

[6] Okatani, T. 2015. *Deep Learning*, Kodansha. (in Japanese)

[7] Albrecht, R. F., Reeves, C. R., and Steele, N. C., eds. 1993. *Artificial Neural Nets and Genetic Algorithms*. Adolf Holzhausens Nachfolger, A-1070 Wien, Austria.

[8] Kamishima, T., Asoh, H., Yasuda, M., Maeda, S., Okanohara, D., Okatani, T., et al. *Deep Learning*. (in Japanese)

[9] Betere, J. I., Kinjo, H., et al. 2018. "Learning Performance of Linear and Exponential Activity Function with Multi-layered Neural Network." *Journal of Electrical Engineering* 6. doi: 10.17265/2328-2223/2018.03.000, USA.

[10] Betere, I. J., Kinjo, H., et al. 2018. "Investigation of Multi-layer Neural Network Performance Evolved by Genetic Algorithms." Artif Life Robotics, doi.org/10.1007/s10015-018-0494-2, Japan.