International Journal of Future Generation Communication and Networking
Vol. 13, No. 4, (2020), pp. 2415–2426

Novel Activation Functions Based on TanhExp Activation Function in Deep Learning

Marina Adriana Mercioni[1] and Stefan Holban[2]
[1,2] *Department of Computer Science*
[1,2] *Politehnica University Timisoara*
[1,2] *Timisoara, Romania*
[1] *marina.mercioni@student.upt.ro*
[2] *stefan.holban@cs.upt.ro*

Abstract

*Our proposal is to make a brief synthesis of the most common and recent activation functions to see how the activation function impacts the performance of a Deep Learning model. For this purpose, we will test popular functions on Deep Learning models and using the obtained information in order to see which activation function is the most suitable in order to increase the performance based on the type of chosen task. We introduce three novel activation functions, that are able to bring performance improvements on object classification tasks using datasets such as MNIST, Fashion-MNIST, CIFAR-10, CIFAR-100, but we will see that we also used a dataset for detection of anomalies in time series. To test them, we used several types of architectures, including LeNet-5 and AlexNet for CIFAR-10 and CIFAR-100, for MNIST and Fashion-MNIST a custom architecture as at the implementation of TanhExp function and a custom architecture for the task of time series. To validate our proposal, we compare them with ReLU, tangent (*tanh*) and TanhExp activation functions.*

Keywords: *activation function, anomaly, CIFAR-10, CIFAR-100, Deep Learning, learnable, ReLU, TanhExp, timeseries*

1. Introduction

Even in 2020 activation functions represent an area of interest due to the role it plays in a neural network. [1] Activation functions power the output of a neuron, map linear data into a nonlinear range, and therefore introduce the non-linearity that the system as a whole need to learn nonlinear data. Along time, in literature, we find different approaches on how to choose an activation function. [2] We cover traditional activation functions like tanh and ReLU, but also the newer one like TanhExp (Tanh Exponential) activation function. Starting from the idea of how the human brain works when it is fed with a lot of information simultaneously, and we just try to understand and to do a binary classification of the information into 'relevant' or 'not-relevant'. 'Not-relevant' information is most of the time just noise, and it affects the learning process. This approach helps us to reduce the high volume of information and we come up with a solution to our task. In the same manner, the activation function works. The activation function helps the neural network use important information and suppress irrelevant data. Another important aspect is the type of function because the neuron cannot learn only with a linear activation function. A non-linear function allows learning from error difference. In our study, we use a traditional activation function, called tangent (tanh) [3], it comes up to solve the disadvantages of sigmoid function [2]. In a recent study, tanh outperformed in the application of Machine Learning to shoreline changes, tanh activation functions are shown to have better overall performance [4].

The paper is structured as follows. Chapter 2 introduces the related work. In Chapter 3 we give a detailed description of our activation functions without and with learnable parameters, which include their definitions, and main properties. In Chapter 4, we prove the efficiency of the functions

ISSN: 2233-7857 IJFGCN

on several datasets from Computer Vision [5][6] to anomalies (outliers) [7] detection in time series. Chapter 5 gives the conclusion of our whole work.

Anomaly data appears in a kind of problems such as bank fraud [8], fraud in healthcare [9], errors in the text [10]. For this purpose, we used a novel technique called variational autoencoders (VAEs) [12][13] to design networks. We used VAEs because this technique is great for learning latent spaces that are well structured, which helps us to have specific directions to encode a meaningful axis of variation in the data. VAE uses the mean and variance parameters to randomly sample one element of the distribution and decodes that element back to the original input (see Figure 1.). The random behavior of this process improves robustness and forces the latent space to encode meaningful representations everywhere: every point sampled in the latent space is decoded to a valid output. [14].

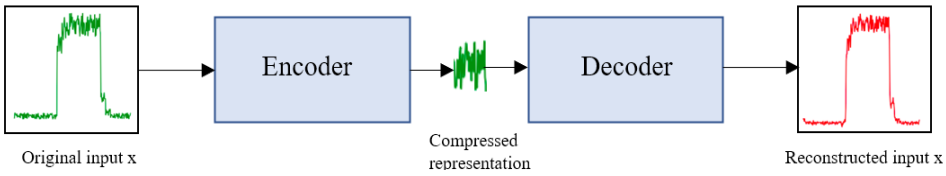

Figure 1. An autoencoder for timeseries

Anomalies are also met like outliers, novelties noise, deviations or exceptions. [11] Over time, many definitions were given to anomaly concepts, but the most appropriate seems to be the definition that tells us that outliers can be seen as observations that do not have the expected behavior. [15] Outlier detection techniques in time series data vary depending on the input data type, the outlier type, and the nature of the method, that we will see in experimental data. Our input time series is a univariate time series (because we have just a single feature).

2. Related work

In this chapter, we present some activation functions which are very popular and we used to compare with our proposals.

2.1. Tanh Activation Function

Tanh or hyperbolic tangent is pretty similar to a sigmoid function, it is a scaled sigmoid function, but its range is from (-1, 1). This means that for any neuron, node, or activation we enter, it will be scaled to a value between -1 and 1. Tanh shape is also sigmoidal (s-shaped). This function ensures a strong mapping between negative and zero inputs that will be around zero in the graph. Other properties are being differentiable and monotonic functions. A strength of tanh function is that its gradient is stronger, which means the derivatives are steeper. But despite its properties, tanh suffers from a vanishing gradient problem. The equation of the sigmoid function is given by:

$$f(x) = \tanh(x) = \frac{1}{1-e^{-x}} \quad (1)$$

2.2. ReLU (Rectified Linear Unit) Activation Function

ReLU activation function is a simple, popular function used frequently in Deep Learning architecture. It is a non-linear function. This function is not bound, its range is [0, +∞). In comparison with sigmoid and tanh that drive to almost all neurons to die at a moment time. But ReLU yields 50% activation of the network, in other words, output 0 for negative inputs, that drive to fewer neurons to be fired (sparse property) and the network becomes lighter. Despite its advantages, ReLU suffers from dying ReLU for activation corresponding to negative inputs. For these activations, the gradient will be equal to 0 because the weights stop to update during descent. But in order to mitigate this issue, Leaky ReLU comes up to fix it. This function just has a small amount called leaky for the negative inputs. This leaky argument is equal to 0.01. The equation of the ReLU activation function is given by:

ISSN: 2233-7857 IJFGCN

$$f(x) = \max(0, x) \tag{2}$$

2.3. TanhExp Activation Function

Tanh Exponential activation function (TanhExp) [1] is a recent activation function, it is characterized by its smoothness and high convergence speed for lightweight neural networks. It is pretty similar to Mish [16] function, another recent activation function. TanhExp is given by:

$$f(x) = x \cdot \tanh(e^x) \tag{3}$$

From its equation, we can see that TanhExp is a combination of functions: ReLU, tanh, and exponential argument.

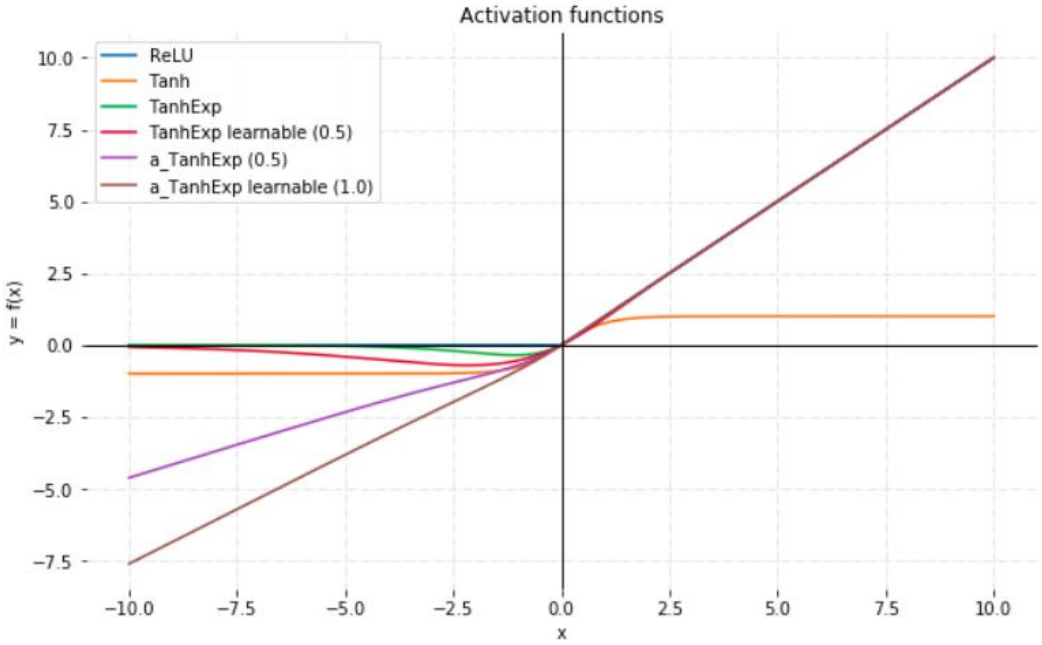

Figure 2. Related work

3. Proposed activation function

As seen in the previous chapter, the TanhExp function was a combination of more functions. Our proposals consist of developing novel functions that derive from TanhExp but the main difference consists of using learnable property that is missing in TanhExp function design.

3.1. TanhExp learnable Activation Function

TanhExp learnable activation function is a smooth function, it is pretty similar to TanhExp, it brings as novelty its learnable parameter that is capable to bring accuracy improvements. Its graphic can be seen in Figure 2. It takes values on both axes, in the range (-∞, +∞). The equation of TanhExp learnable function is given by:

$$f(x) = x \cdot \tanh(e^{\alpha \cdot x}) \tag{4}$$

Where α is a uniparameter that can be predefined or learnable.

In Figure 2. we can see TanhExp learnable highlighted in red for α = 0.5, overlapped on TanhExp function highlighted in green.

The design of the TanhExp learnable function was inspired by the TanhExp function. We implemented the function in Tensorflow 2.2.4, using Keras API.

This proposal benefits from the TanhExp properties, being a continuous, monotonous, unbounded above and bounded below function. To validate this function, we used several architectures and mapping to different types of tasks.

3.2. a_TanhExp Activation Function

a_TanhExp activation function is a parametric activation function inspired from the TanhExp activation function. The parameter *a* controls the concavity of the global minima of the activation function where $a = 0$ is just the TanhExp activation function. Varying *a* in negative scale reduces the concavity and on the positive scale magnify the concavity. *a* is introduced to fight gradient death scenarios due to the sharp global minima of the TanhExp activation function. Its equation is given by:

$$f(x) = x \cdot \tanh(e^x + a) \tag{5}$$

3.3. a_TanhExp learnable Activation Function

a_TanhExp learnable activation function is pretty similar to a_TanhExp activation function. In this case, the parameter *a* becomes a learnable parameter.

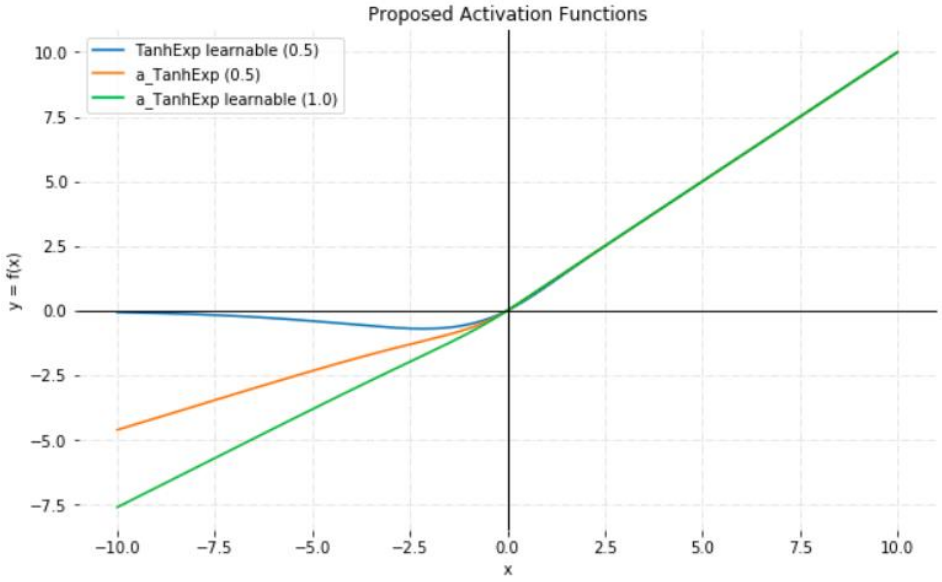

Figure 3. Proposed functions

4. Experimental results

We compare our proposals with tanh, ReLU, and TanhExp activation functions. We will use LeNet-5 architecture [17], AlexNet [18], and custom architectures.

4.1. Experiment no. 1 MNIST Dataset

As architecture, we used like the TanhExp activation function case, the basic network with 15 layers, that we trained on MNIST [19] dataset for 10 epochs. Other information, we mention validation split = 0.20, batch size = 64.

Table 1. Validation metrics for MNIST dataset

Activation function	Validation Accuracy	Validation Loss
ReLU	98.61%	0.0566
tanh	98.33%	0.0590
TanhExp	98.60%	0.0609
TanhExp learnable	98.47%	**0.0547**
a_TanhExp	98.18%	0.0647

ISSN: 2233-7857 IJFGCN

a_TanhExp learnable	**98.62%**	0.0593

In Table 1 we can see that our proposal is giving the best accuracy on MNIST dataset.

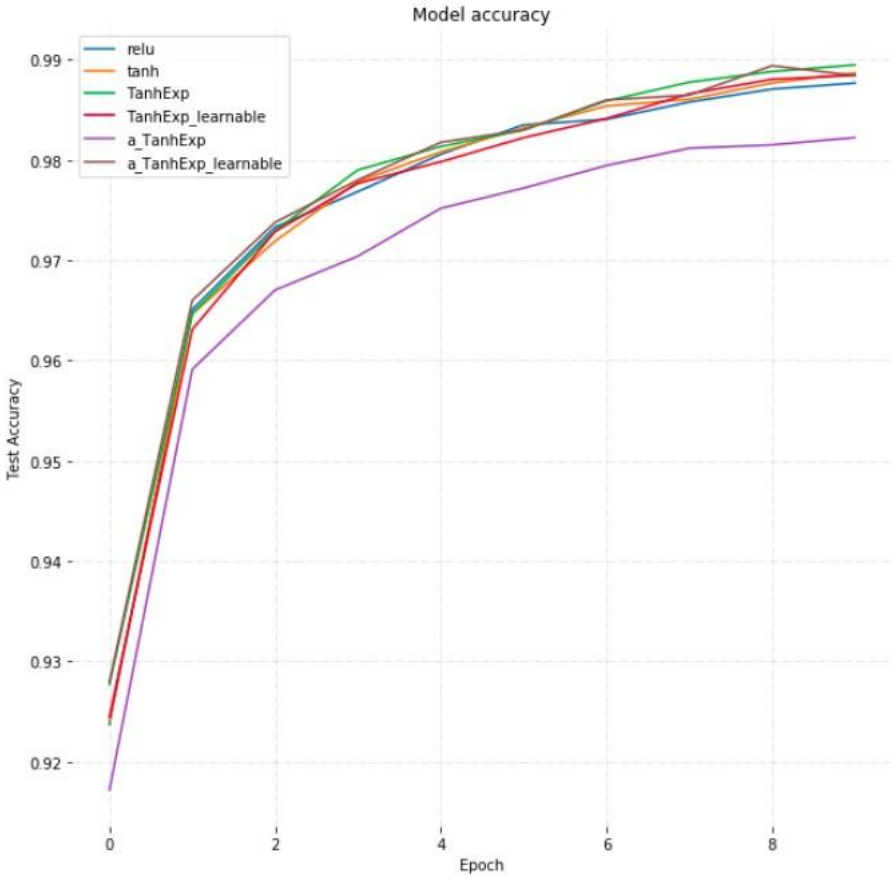

Figure 4. Validation accuracy on MNIST dataset

ISSN: 2233-7857 IJFGCN

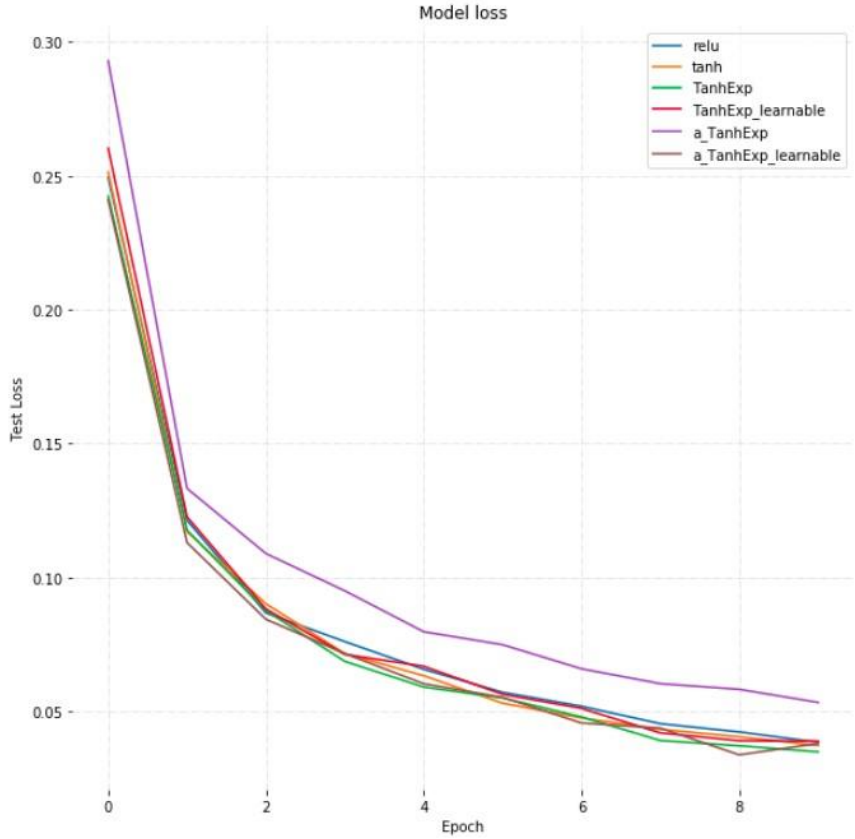

Figure 5. Validation loss on MNIST dataset

4.2. Experiment no. 2 Fashion-MNIST Dataset

As architecture, we used like the MNIST dataset case, the basic network with 15 layers, we just trained on Fashion-MNIST [20] dataset for 20 epochs, batch size = 64, and validation split = 0.2.

Table 2. Validation metrics for MNIST dataset

Activation function	Validation Accuracy	Validation Loss
ReLU	91.59%	0.2736
tanh	91.33%	0.3287
TanhExp	91.44%	0.3392
TanhExp learnable	82.50%	0.5243
a_TanhExp	91.53%	0.3353
a_TanhExp learnable	**91.63%**	0.3154

ISSN: 2233-7857 IJFGCN

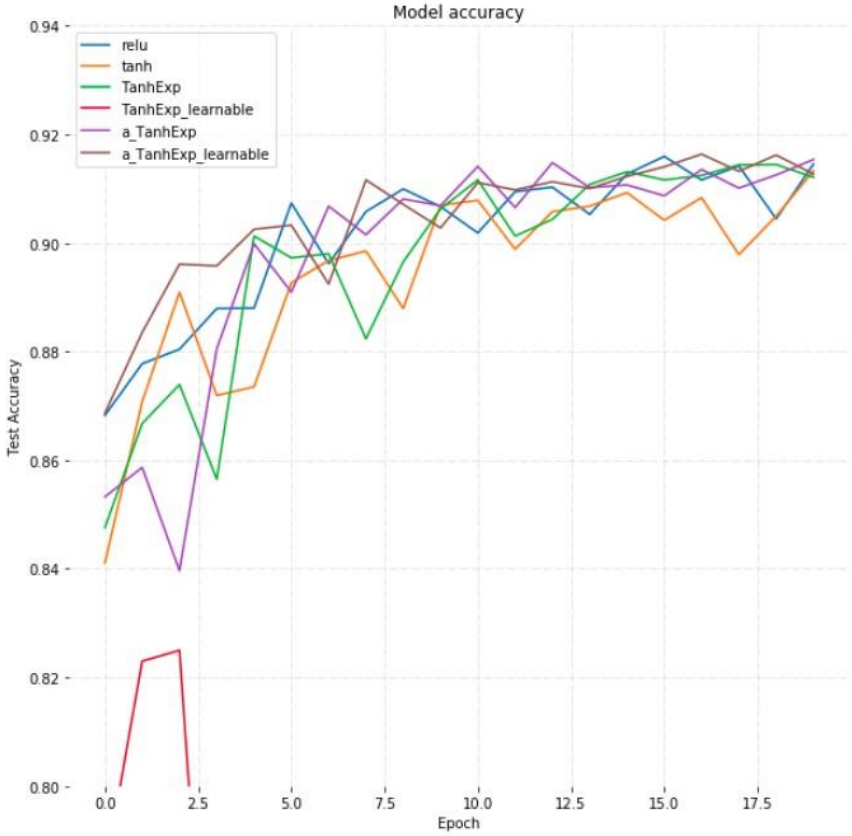

Figure 6. Validation accuracy on the Fashion-MNIST dataset

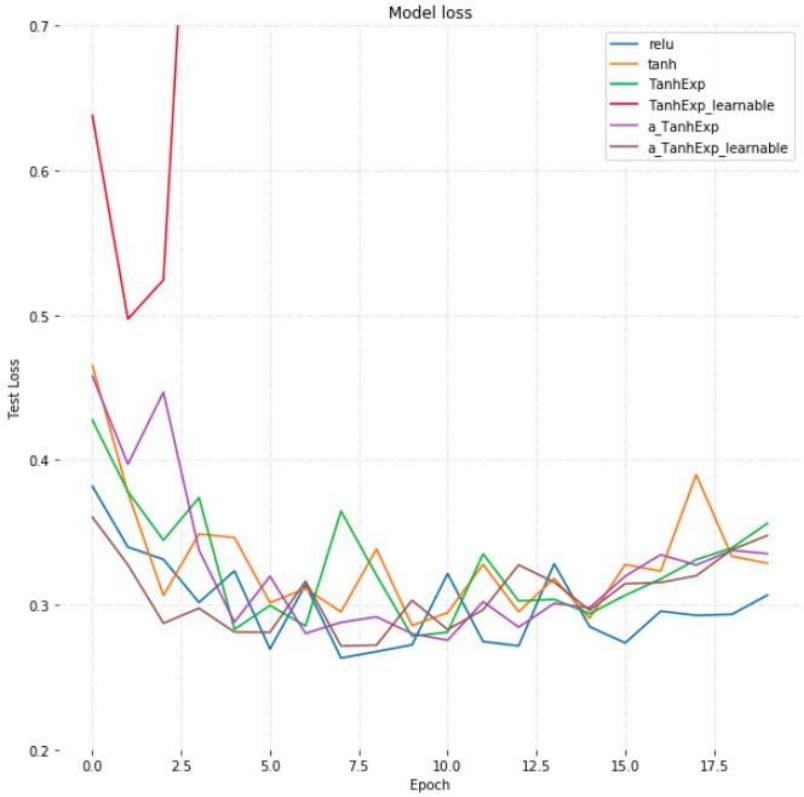

Figure 7. Validation loss on the Fashion-MNIST dataset

ISSN: 2233-7857 IJFGCN

4.3. Experiment no. 3 CIFAR-10 and CIFAR-100 Datasets

As architectures, we used LeNet-5 and AlexNet, we just trained for 20 epochs for each architecture. Other information, we mention that in the case of LeNet-5 architecture, we used Batch Normalization technique [21], batch size = 128 and Data Augmentation technique [22], we applied randomly shift images horizontally (width_shift_range=0.1), randomly shift images vertically (height_shift_range=0.1), we set mode for filling points outside the input boundaries 'nearest', set randomly flip images (horizontal_flip=True), and validation_split=0.2. In the case of AlexNet architecture, we used Dropout technique [23] (dropout rate = 0.5) on 6- and 7-layer positions, as optimizer 'rmsprop', as loss function 'sparse_categorical_crossentropy', Batch Normalization technique, batch size = 128 and Data Augmentation technique just like in the case of LeNet-5 architecture.

Table 3. Validation accuracy for CIFAR-10 dataset

Activation function	LeNet-5	AlexNet
ReLU	70.03%	83.32%
tanh	64.91%	75.43%
TanhExp	69.05%	83.98%
TanhExp learnable	44.21%	79.28%
a_TanhExp	69.74%	**84.97%**
a_TanhExp learnable	**70.28%**	83.19%

In Table 3, we can see that our proposal gives the best accuracy. So, it makes it to be a powerful and robust solution in comparison with ReLU, tanh, and TanhExp functions.

Table 4. Validation accuracy for CIFAR-100 dataset

Activation function	LeNet-5	AlexNet
ReLU	39.81%	60.62%
tanh	37.49%	53.57%
TanhExp	40.13%	60.99%
TanhExp learnable	40.58%	56.47%
a_TanhExp	37.70%	61.07%
a_TanhExp learnable	**40.55%**	**61.36%**

4.4. Experiment no. 4 Detect anomalies in time series data

In this sub-chapter, we considered the Numenta Anomaly Benchmark (NAB) [24]. It provides artificial time series data containing labeled anomalous periods of behavior. Data are ordered, timestamped, single-valued metrics. We used the `art_daily_small_noise.csv` file for training and the `art_daily_jumpsdown.csv` file for testing. This dataset is univariate, it has only one feature and it allowed us to prove us anomaly detection effectively. As we said in the introduction, anomalies are met when the data points are outliers or an exceptional event happened. Time series is the data captured on a fixed timestamp when we analyze it seems to be like a trend or seasonality. Identifying anomalies in time series is challenging, depending on the input data type, the outlier type, and the nature of the method. [15] We normalized data values from the training time series. We have data for every 5 minutes for 14 days, so we have 288-timestamps per day. We used an existing custom convolutional reconstruction autoencoder [25-27] model, we did not insist too much on architecture because our objective is to find a suitable activation function. The model has an input of shape (batch size, sequence's length, number of features) and returns the output of the same shape. In our case, sequence length is 288, and number of features is 1.

ISSN: 2233-7857 IJFGCN

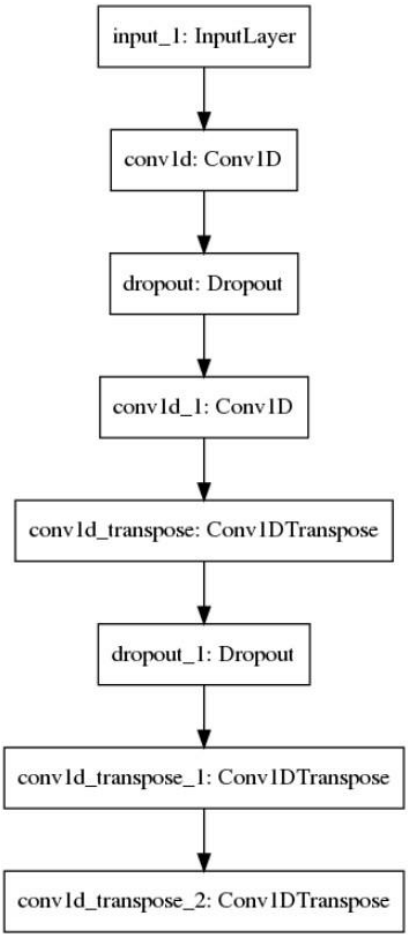

Figure 8. Time series convolutional reconstruction autoencoder architecture

The convolutional reconstruction autoencoder is composed of two Conv1D layers (20 and 16 filters) and Dropout (dropout rate = 0.25) after each Conv1D layer. For reconstruction, we have three Conv1DTranspose layers (16, 20, and 1 filters).

In order to detect the anomalies in our time series data we just determined how our model works in input data reconstruction. For this purpose, we tried to minimize the error between predicted data and training input data. We computed MAE [28] (*mean absolute error*) loss on training samples.

$$MAE = \frac{\sum_{i=1}^{n} | y_i - x_i |}{n} \quad (6)$$

Where y_i is predicted corresponding to the reconstructed sample, and x_i that corresponds to the original sample.

After that, we took the maximum value of the MAE loss value. This value corresponds to the worst case of how our model performed the reconstruction of a sample. We considered this value as a threshold for our anomalies. In other words, in the case of a reconstruction loss for a sample is greater than this established value, threshold, then we can say that the model meets a pattern that he doesn't know, and this sample is considered as an anomaly. We trained 50 epochs for each activation function separately. We used as optimizer Adam (learning rate = 0.001) and loss function "mse" (*mean squared error*), batch size = 128, and a validation split = 0.2.

Table 5. Validation loss for time series data

ISSN: 2233-7857 IJFGCN

Activation function	Validation loss
ReLU	0.0054
tanh	0.0024
TanhExp	0.0026
TanhExp learnable	0.0054
a_TanhExp	**0.0017**
a_TanhExp learnable	0.0023

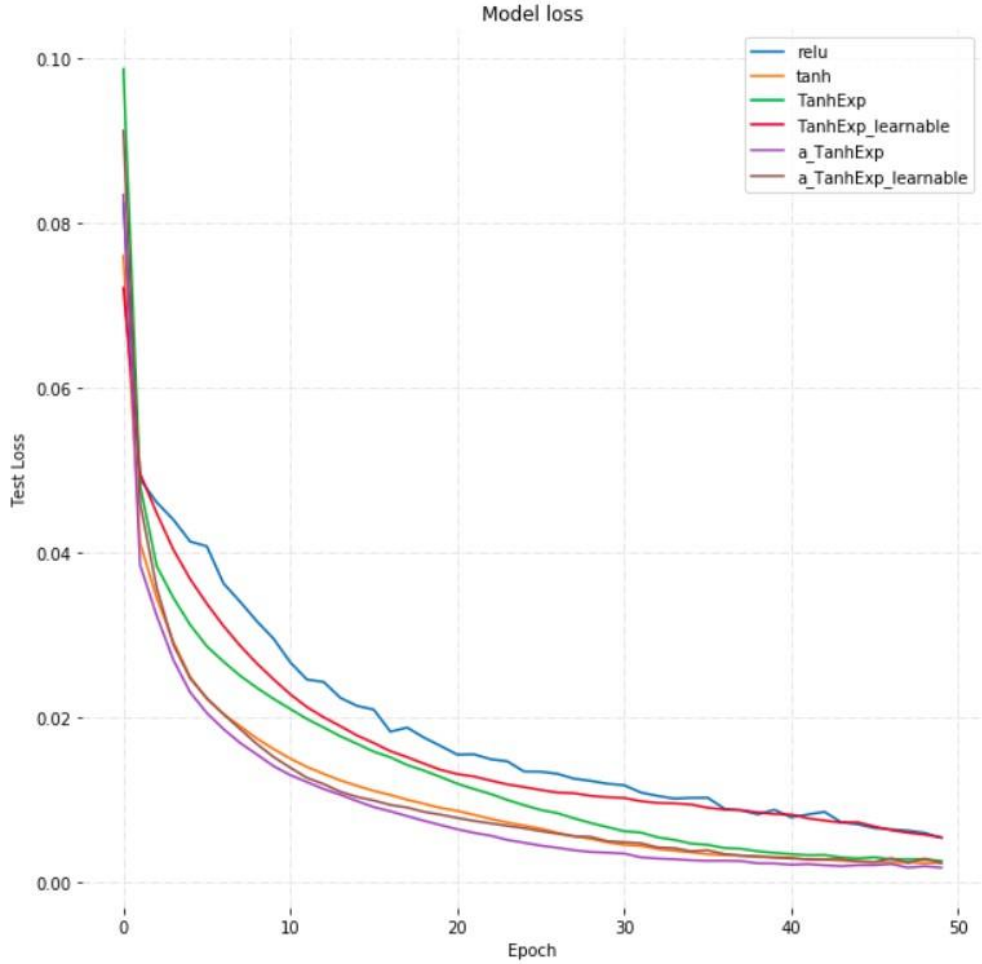

Figure 9. Time series test data loss

5. Conclusions

In this paper, we proposed three novel activation functions called *TanhExp learnable*, *a_TanhExp*, *a_TanhExp learnable*. These functions bring improvements of performance on variate datasets varying the type of task in Deep Learning. Their simplicity and properties make them meaningful bringing competitive accuracy in Computer Vision tasks and anomalies detection of time series data task. The experimental data strength our beliefs that these functions can be used successfully in this kind of tasks. Also, they are a robust solution for lightweight neural networks. For future work, we would like to test them on other datasets and other types of architectures.

References

[1] X. Liu and X. Di, "TanhExp: A Smooth Activation Function with High Convergence Speed for Lightweight Neural Networks", (**2020**).

ISSN: 2233-7857 IJFGCN

[2] S. Narayan, "The generalized sigmoid activation function: Competitive supervised learning", (**1997**).

[3] Y. LeCun, Y. Bengio and G. Hinton, "Deep learning", Nature, (**2015**).

[4] Je-Chian Chen and Yu-Min Wang, "Comparing Activation Functions in Modeling Shoreline Variation Using Multilayer Perceptron Neural Network", Water, 12, 1281; doi:10.3390/w12051281, (**2020**).

[5] Dana H. Ballard and Christopher M. Brown, "Computer Vision", (**1982**).

[6] Huang, T., "Computer Vision: Evolution and Promise", CERN, (**1996**).

[7] Zimek, Arthur and Schubert, Erich, "Outlier Detection, Encyclopedia of Database Systems", Springer New York, (**2017**).

[8] Vaishnavi Nath Dornadula and S. Geetha, "Credit Card Fraud Detection using Machine Learning Algorithms", Procedia Computer Science Vol. 165, (**2019**).

[9] D. Thornton, M. Brinkhuis, C. Amrit and R. Aly, "Categorizing and Describing the Types of Fraud in Healthcare", Procedia Computer Science Volume 64, pp. 713-720, (**2015**).

[10] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu and J. Gao, "Deep Learning Based Text Classification: A Comprehensive Review", arXiv:2004.03705v1 [cs.CL], (**2020**).

[11] V. J. Hodge, and J. Austin, "A Survey of Outlier Detection Methodologies", Artificial Intelligence Review, (**2004**).

[12] M. Kramer, "Nonlinear principal component analysis using autoassociative neural networks", AIChE Journal. 37, (**1991**).

[13] P. K. Diederik and M. Welling, "Auto-Encoding Variational Bayes", arXiv https://arxiv.org/abs/1312.6114, (**2013**).

[14] F. Chollet, "Deep Learning with Python", Manning, (**2018**).

[15] A. B. Garcia and A. Conde, "A review on outlier/anomaly detection in time series data", arXiv:2002.04236v1, (**2020**).

[16] D. Misra, "Mish: A Self Regularized Non-Monotonic Neural Activation Function", (**2019**).

[17] Y. LeCun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-Based Learning Applied to Document Recognition", (**1998**).

[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, "ImageNet Classification with Deep Convolutional", 2012.

[19] Y. LeCun, and C. Cortes, "MNIST handwritten digit database", (**2010**).

[20] H. Xiao, K. Rasul, R. Vollgraf, "Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms", (**2017**).

[21] S. Ioffe and C. Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", (**2015**).

[22] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning", arXiv:1712.04621 [cs.CV], (**2017**).

[23] N. Srivastava, G. Hintonand, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting", (**2014**).

[24] Numenta Anomaly Benchmark (NAB), https://www.kaggle.com/boltzmannbrain/nab, (**2015**).

[25] I. Goodfellow, Y. Bengio and A. Courville, "Deep Learning". MIT Press. ISBN, (**2016**).

ISSN: 2233-7857 IJFGCN

[26] P. Vincent and H. Larochelle, "Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion". Journal of Machine Learning Research. 11: 3371–3408, (**2010**).

[27] Anomaly detection in time series, https://github.com/pavithrasv, (**2020**).

ISSN: 2233-7857 IJFGCN