P-TELU : Parametric Tan Hyperbolic Linear Unit Activation for Deep Neural Networks

Rahul Duggal
rahulduggal2608@gmail.com

Anubha Gupta
anubha@iiitd.ac.in

SBILab (http://sbilab.iiitd.edu.in/)
Deptt. of Electronics and Communication Engineering
Indraprastha Institute of Information Technology - Delhi (IIIT-D), India

Abstract

This paper proposes a new activation function, namely, Parametric Tan Hyperbolic Linear Unit (P-TELU) for deep neural networks. The work is inspired from two recently proposed functions: Parametric RELU (P-RELU) and Exponential Linear Unit (ELU). The specific design of P-TELU allows it to leverage two advantages: (1) the flexibility of tuning parameters from the data distribution similar to P-RELU and (2) better noise robustness similar to ELU. Owing to larger gradient and early saturation of tan hyperbolic compared to exponential function, the proposed activation allows a neuron to reach/exit from the noise robust deactivation state earlier and faster. The performance of the proposed function is evaluated on CIFAR10 and CIFAR100 image dataset using two convolutional neural network (CNN) architectures : KerasNet, a small 6 layer CNN model, and on 76 layer deep ResNet architecture. Results demonstrate enhanced performance of the proposed activation function in comparison to the existing activation functions.

1. Introduction

Deep neural networks, particularly, deep convolutional neural networks (CNN) have set new benchmarks for many computer vision problems [8, 3]. This is attributed to the hierarchical learning representation that allows learning of abstract concepts [18]. Generally, deeper networks are observed to learn better representations [8, 13, 15]. The increase in depth has been aided by several advances in the field including better regularization strategies such as Dropout [14] and Batch Normalization [4], better optimizers such as Adam [6] and RMSProp [16] and, better initializations including [12, 2] among others. Newer constructs with inception modules [15] and identity connections [3] have also significantly increased the upper bound of stackable layers. Recently, a great amount of attention has also been paid to the activation functions used in these networks.

Activation functions are a critical component of a CNN. They introduce non-linearity in the model that allows it to learn complex decision boundaries. Several activation functions have been proposed in the past including RELU, P-RELU, ELU, etc. [8, 1, 2, 11, 7, 5]. These functions are designed to overcome the problem of exploding/vanishing gradients that is commonly encountered with traditional functions such as the sigmoid. Further, it has been shown that a mean closer to zero speeds up the convergence of the network while training [9, 1]. Accordingly, recent activation functions have tried to incorporate this fact through specific design choices. For example, Leaky-Relu [10] implements a small, constant, positive sloped straight line for negative inputs. Similarly, the Exponential Linear Unit [1] implements the exponential function for negative inputs. Recently, it has been shown [2] that the constant slope in the case of Leaky-Relu can be parametrized in terms of learnable weights, with little additional computational cost while adding considerable flexibility to the activation function. Inspired by the above works on parametrization and design choices of activation functions, we propose a new activation function, Parametric Tan Hyperbolic Linear Unit (P-TELU), for deep neural networks.

The organization of the paper is as follows: Section 2 briefly reviews some of the popular activation functions. Section 3 presents the proposed P-TELU activation function. In Section 4, the proposed activation function is benchmarked against some of the recent activation functions on two standard CNN architectures. Section 5 presents some conclusions and suggestions for future improvements.

2. Background

A lot can be learned by reviewing the evolution of activation functions. Traditionally, sigmoid function defined as $\sigma : \mathbb{R} \rightarrow [0, 1], z \rightarrow \frac{1}{1+e^{-z}}$, has been the most widely

used activation function. A contributing factor in its popularity is the direct probabilistic interpretation of its output range $[0, 1]$. However, as discussed previously, the non zero output mean negatively impacts the convergence of neural networks. This fact led to the shift from sigmoid to the tan hyperbolic for use as an activation function because the tanh function has range $[-1, 1]$, in contrast to the sigmoid which has a range $[0, 1]$. Recently, other functions have been proposed that provide better performance in terms of speedy convergence of deep neural networks. Some of the significant ones are reviewed below.

RELU [8]: The rectified linear exponential unit activation is defined as:

$$f(z) = \begin{cases} z & z > 0 \\ 0 & z \leq 0. \end{cases} \tag{1}$$

RELU has been shown to speed up the training process over tanh by 6 times. This function uses the identity function for positive inputs and hence, has a gradient of 1 for $z > 0$. This feature allows it to address the problem of vanishing/exploding gradient in deep neural networks. However, a major disadvantage of RELU is the zero gradient for negative inputs. Thus, if a neuron encounters a negative input, it enters a de-activated state from where it can never exit. This can lead to many de-activated neurons in the network which is a computation and memory expensive waste of model capacity.

L-RELU [10] : The leaky RELU addresses the problem of de-activated neurons by introducing a linear function for negative inputs. This affords a small gradient for negative inputs and thus allows some of the de-activated neurons to re-activate. L-RELU is defined as below:

$$f(z) = \begin{cases} z & z > 0 \\ \alpha z & z \leq 0, \end{cases} \tag{2}$$

where α is a small fixed positive constant.

P-RELU [2]: In L-RELU, α was a fixed constant. P-RELU proposed to learn the optimal value of α through back propagation, from the data distribution itself. This affords a more general rectifier function at negligible additional computational cost. The number of additional learnable parameters are further reduced by sharing the $\alpha's$ across a channel.

ELU [1]: The exponential linear unit modifies (1) for negative inputs as below:

$$f(z) = \begin{cases} z & z > 0 \\ \alpha(e^z - 1) & z \leq 0. \end{cases} \tag{3}$$

where α is a constant that is generally set to 1. The exponential function in ELU provides a noise robust deactivation state for large negative inputs. It was shown empirically that this resulted in significant gains over activation functions such as RELU and L-RELU which implement a straight line in the negative half of the input plane.

3. Proposed P-TELU Activation Function

Motivated with the designs presented above that show improved performance in image processing tasks such as image classification, we propose a new activation function for deep neural networks. We build upon ideas inspired from the choices of activation functions described in the previous section. Firstly, similar to RELU, Leaky-RELU, P-RELU and ELU, we use identity mapping for positive inputs. This ensures that the gradient for positive inputs is '1' and hence, takes care of the problem of exploding/vanishing gradients. Secondly, for the negative inputs, we use tan hyperbolic function that is somewhat similar to exponential function of ELU. This provides a noise robust de-activation state. The proposed activation function is defined as:

$$f(x) = \begin{cases} z & z > 0 \\ \alpha \times tanh(\beta \times z) & z \leq 0, \alpha \geq 0, \beta \geq 0 \end{cases} \tag{4}$$

where the parameters α and β are learned along with the filter weights from the training data. We have constrained these parameters to remain non-negative. This ensures that P-TELU performs atleast as good as RELU and in the worst case will degenerate to RELU. Similar to P-RELU [2], we have not used any regularization on α and β since that may degenerate P-TELU to RELU.

Geometrically, α controls the saturation value and β controls the convergence rate. The choice of using the $tanh$ function for the negative inputs instead of the exponential function as in ELU is guided by the intuition that $tanh$ has 1) a higher gradient for small negative inputs and 2) saturates earlier than $e^x - 1$ as is evident from fig 1. Thus it reaches the noise robust deactivation state earlier and faster. The larger gradients also allow it to exit from saturation at a faster rate.

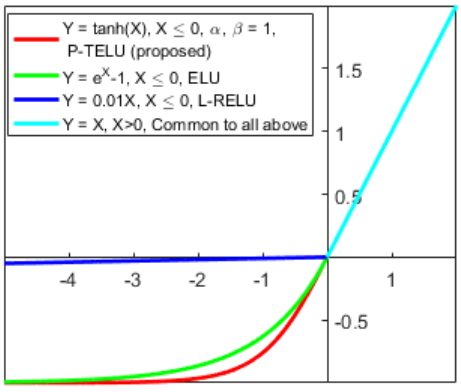


Figure 1: Plots of various activation functions

On the computational front, the number of additional

learnable parameters in P-TELU are twice compared to P-RELU. This number is still insignificant compared to the total number of weights learned in the Convolutional Neural Network model. For example, in the 76 layer ResNet considered in this paper, 3968 additional parameters have been introduced by P-TELU over and above 4,53,002 parameters required to be learned in ResNet with RELU activation function. This shows that there is no significant increase in the number of parameters required to be learned with the proposed P-TELU. Thus, the gain in accuracy of the network, as shown in the next section, is not due to the enhanced model capacity, but because of the better representation afforded by the flexibility of the learnt parametrization.

To find the optimal α and β through backpropagation, we require to compute the derivative of the output loss function L with respect to α and β.This is computed as below.

$$\frac{\mathrm{d}L}{\mathrm{d}\alpha_{i,j}} = \frac{\mathrm{d}L}{\mathrm{d}f_{i,j}} \times \frac{\mathrm{d}f_{i,j}}{\mathrm{d}\alpha_{i,j}} \tag{5}$$

$$\frac{\mathrm{d}L}{\mathrm{d}\beta_{i,j}} = \frac{\mathrm{d}L}{\mathrm{d}f_{i,j}} \times \frac{\mathrm{d}f_{i,j}}{\mathrm{d}\beta_{i,j}} \tag{6}$$

$$\frac{\mathrm{d}f_{i,j}}{\mathrm{d}\alpha_{i,j}} = \begin{cases} 0 & z_{i,j} > 0 \\ tanh(\beta_{i,j} \times z_{i,j}) & z_{i,j} \leq 0 \end{cases} \tag{7}$$

$$\frac{\mathrm{d}f_{i,j}}{\mathrm{d}\beta_{i,j}} = \begin{cases} 0 & z_{i,j} > 0 \\ \alpha \times z_{i,j} \times (1 - tanh^2(\beta_{i,j} \times z_{i,j})) & z_{i,j} \leq 0 \end{cases} \tag{8}$$

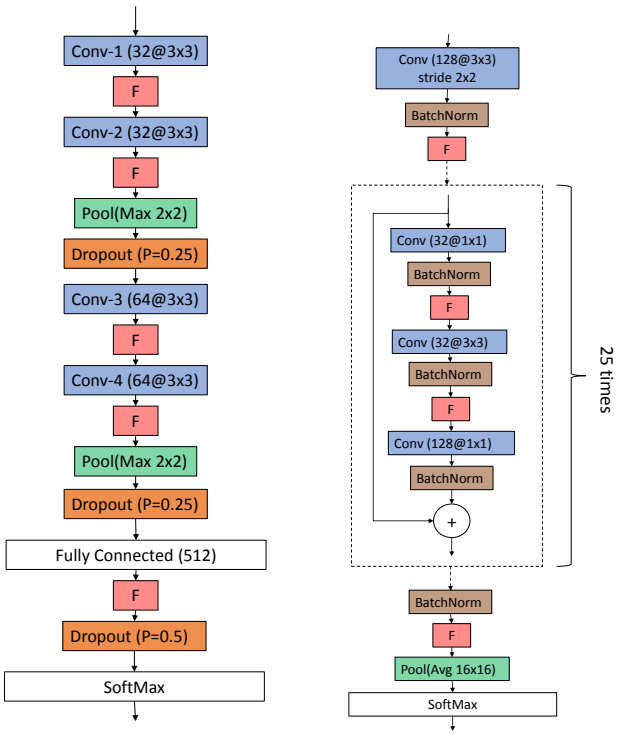
where subscripts (i, j) denote the j^{th} neuron in layer i, $z_{i,j}$ refers to the net input of the neuron at position (i, j), and $f_{i,j}$ refers to the output of the neuron after applying the activation.

4. Experiments using P-TELU

In this section, we benchmark our activation function with RELU, P-RELU and ELU using two networks: a relatively shallow (6 layer KerasNet) and a deep residual network (76 layers). The performance has been evaluated on two standard image datasets described below. Simulations of this work were carried using Keras configured with Theano backend on a Ubuntu 14.04 system with 2 x P5000 GPUs with 32GB total GPU memory.

4.1. Dataset

We report performance on CIFAR10 and CIFAR100 datasets. These datasets consist of 60,000 RGB images of size 32×32 belonging to 10 and 100 classes, respectively. As standard procedure, we train the networks on the training set consisting of 50,000 images and evaluate performance on the test set consisting of 10,000 images. For data augmentation, we performed random horizontal shifting by 3 pixels as well as horizontal flipping. Mean subtraction and



(a) KerasNet (b) ResNet

Figure 2: Architectures of the networks used in this paper

Scaling by 255 is carried out to bring the image intensities in the range $[-0.5, 0.5]$.

4.2. Experiment 1: Benchmarking on KerasNet

We evaluate performance of various activation functions on a standard architecture provided in Keras Python Library. This network is also referred as KerasNet [17] and consists of 6 layers. The architecture is described in Figure 2. The red coloured F boxes refer to the activation functions. For every activation function, we train the corresponding model using stochastic gradient descent (SGD) algorithm with a learning rate of 0.01 and momentum of 0.9. Both the parameters α and β of P-TELU required to be learned are initialized at 1. The α parameter of ELU is fixed to 1, while the parameters of P-RELU are initialized to zero. Best accuracies are obtained over 300 epochs on both CIFAR10 and CIFAR100 dataset and are summarized in Table 1. From Figure 3, it is observed that P-TELU leads to both faster gain in accuracy and higher accuracy at convergence.

4.3. Experiment 2: Benchmarking on ResNet

In this subsection, we present the performance evaluation of various activation functions on 76 layer deep ResNet architecture. The network is trained using Adam optimizer

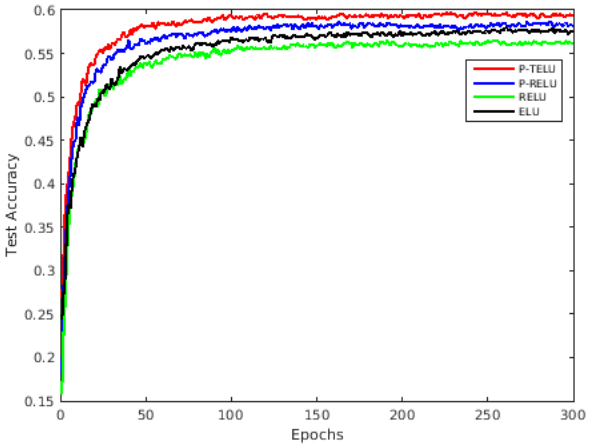


Figure 3: Accuracy v/s epochs curve for KerasNet on CIFAR100 dataset

Table 1: KerasNet: Best accuracy observed over 300 epochs

Activation Function	CIFAR-10	CIFAR-100
RELU	85.45	56.56
ELU	85.84	57.97
P-RELU	86.26	58.75
P-TELU (Proposed)	**86.5**	**59.76**

with default parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$ [6]. The learning rate is set at epochs 1, 20, 40, 60 and 80 as $10^{-3}, 5 \times 10^{-4}, 10^{-4}, 5 \times 10^{-4}$ and 10^{-5}, respectively. We imposed L^2 regularization of 10^{-4} on all learnable parameters excluding α and β. Two experiments have been conducted: one with non-negative constraints on α and β and second with a constraint that $\alpha, \beta \geq 0.01$. In the worst case, the first set of constraints will degenerate P-TELU to RELU, while the second set of constraints will degenerate P-TELU to Leaky-RELU because $tanh(x) \approx x$ for small x. The best accuracy are obtained over 100 epochs on both CIFAR10 and CIFAR100 dataset and are summarized in Table 2.

Table 2: ResNet: Best accuracy observed over 100 epochs

Activation Function	CIFAR10	CIFAR100
RELU	90.77	70.06
ELU	91.26	69.05
P-RELU	90.99	69.05
P-TELU with $\alpha, \beta \geq 0$	**91.52**	70.13
P-TELU with $\alpha, \beta \geq 0.01$	91.16	**70.63**

5. Discussion

In order to observe the optimally learned values of α and β through backpropagation, we plot the layer-wise average for ResNet in figure 4. The layer-wise averaging is calculated as below :

$$\alpha_i{}^{avg} = \frac{\sum_j \alpha_{i,j}}{\sum j}, \tag{9}$$

$$\beta_i{}^{avg} = \frac{\sum_j \beta_{i,j}}{\sum j}, \tag{10}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ refer to the value of α and β for the j^{th} neuron in layer i.

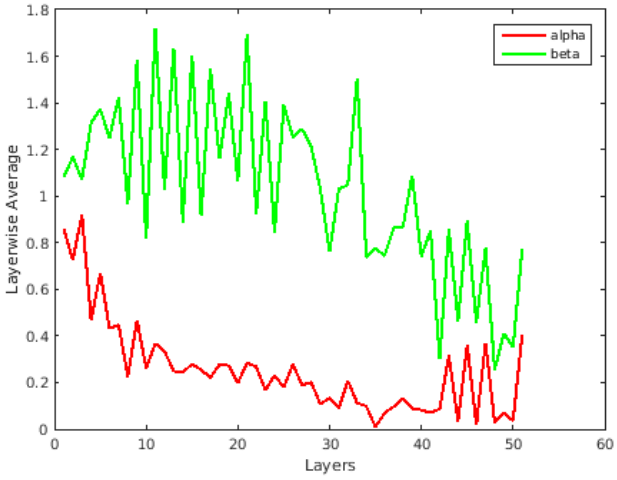


Figure 4: Layerwise averaged values of α and β ($\alpha, \beta \geq 0$), calculated using (10), on the 76 layer ResNet model fitted with P-TELU activation trained on the CIFAR10 dataset.

Figure 4 offers an interesting observation wherein average values of α and β decrease as we progress deeper into the CNN. This finding is consistent with [2], where a similar decrease was observed for the P-RELU activation. It is a known fact that filters in the earlier layers of a CNN capture lower level information such as color blobs and edges [8, 18, 2]. In order to maximize this information capture, the activation function allows even the negative inputs to flow to the output with less attenuation and hence, these layers learn higher αs. With the earlier layers capturing increased coarser level information, the model capacity of the deeper layers becomes available for learning discriminative features that may increase the accuracy of the model by better adapting to the non-linear decision boundaries. Hence, appropriate lower or moderate values of αs are learned in the deeper layers as apparent from Figure-4 that impart more non-linearity to the activation function, add discriminativeness to the model, and enable the model to learn complex decision boundaries. In summary, the adaptable parameters alpha and beta serve different purposes for different layers of the CNN. Optimally learning them from the data leads to early rise in accuracy as well as a higher overall accuracy of CNN in comparison to CNN's trained with previously proposed activation functions. (Figure-3).

6. Conclusion

In this paper, we have presented a new activation function, namely, P-TELU for deep neural networks and have evaluated its performance over two standard convolutional neural networks: the 6 layer KerasNet CNN model and the 76 layer CNN with ResNet architecture. The performance has been evaluated in terms of classification accuracy over two standard image classification datasets: CIFAR 10 and CIFAR 100. The proposed function performs better compared to existing activation functions RELU, ELU, and P-RELU. Moreover, P-TELU achieves faster convergence. In future, we will evaluate its performance on other type of deep networks such as AutoEncoders, DenseNets and with other type of problems such as segmentation and localization.

7. Acknowledgments

Authors gratefully acknowledge the research funding support (Grant Number: 1(7)/2014-ME&HI) from the Ministry of Communication and IT, Govt. of India for this research work.

References

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

[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*, pages 1026–1034, 2015.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[4] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[5] X. Jin, C. Xu, J. Feng, Y. Wei, J. Xiong, and S. Yan. Deep learning with s-shaped rectified linear activation units. *arXiv preprint arXiv:1512.07030*, 2015.

[6] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] K. Konda, R. Memisevic, and D. Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *arXiv preprint arXiv:1402.3337*, 2014.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[9] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller. Efficient backprop. In *Neural networks: Tricks of the trade*, pages 9–48. Springer, 2012.

[10] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, volume 30, 2013.

[11] J. M. McFarland, Y. Cui, and D. A. Butts. Inferring nonlinear neuronal computation based on physiologically plausible inputs. *PLoS Comput Biol*, 9(7):e1003143, 2013.

[12] D. Mishkin and J. Matas. All you need is a good init. *arXiv preprint arXiv:1511.06422*, 2015.

[13] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[14] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[16] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2), 2012.

[17] P. Veličković, D. Wang, N. D. Lane, and P. Liò. X-cnn: Cross-modal convolutional neural networks for sparse datasets. *arXiv preprint arXiv:1610.00163*, 2016.

[18] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.