SigmoReLU: An improvement activation function by combining Sigmoid and ReLU

Stamatis Mastromichalakis[1]

[1]London South Bank University / IST College,
Pireos 72, GR-18346, Moschato, Athens, Greece
Email: stamatis@tmnetworks.gr

ABSTRACT

Two of the most common activation functions (AF) in deep neural networks (DNN) training are Sigmoid and ReLU. Sigmoid was tend to be more popular the previous decades, but it was suffering with the common vanishing gradient problems. ReLU has resolved these problems by using zero gradient and not tiny values for negative weights and the value "1" for all positives. Although it significant resolves the vanishing of the gradients, it poses new issues with dying neurons of the zero values. Recent approaches for improvements are in a similar direction by just proposing variations of the AF, such as Leaky ReLU (LReLU), while maintaining the solution within the same unresolved gradient problems. In this paper, the combining of the Sigmoid and ReLU in one single function is proposed, as a way to take the advantages of the two. The experimental results demonstrate that by using the ReLU's gradient solution on positive weights, and Sigmoid's gradient solution on negatives, has a significant improvement on performance of training Neural Networks on image classification of diseases such as COVID-19, text and tabular data classification tasks on five different datasets.

MSC Subject Classification: 68T07, 68T45, 68T10, 68T50, 68U35
Keywords: Activation Function Combination, dying / vanishing gradients, ReLU, Sigmoid, Neural Networks, Keras, Medical Image Classification

1. INTRODUCTION

In previous decades, neural networks have usually employed logistic sigmoid activation functions. Unfortunately, this type of AF is affected by saturation issues such as vanishing gradient. To overcome such weakness and improve accuracy results, an active area of research is trying design novel activation functions (Franco Manessi et al., 2019), with the ReLU appears be the most well-established the last years. However, ReLU also suffers from 'dying gradient' problem and is has slightly impact on training. Many variations of the AF, such as LReLU are proposed, to solve this issue, while maintaining the solution within the same unresolved gradient problems. Although recent developments of AFs for Shallow and Deep Learning Neural Networks (NN), such as the QReLU/m-QReLU (Parisi et al., 2020a), m-arcsinh (Parisi et al., 2020b), and ALReLU (Mastromichalakis, 2020) the repeatable and reproducible functions have remained very limited and confined to three activation functions regarded as 'gold standard' (Parisi et al., 2020b). The sigmoid and tanh are well-known for their common vanishing gradient issues and only ReLU function seems to be more accurate and scalable for DNNs, despite the 'dying ReLU' problem.

In this work, a new AF, SigmoReLU, is proposed, by combining the advantages of the two different state-of-the-art AFs, ReLU and Sigmoid. This new AF is using both gradient solutions of ReLU and Sigmoid, depending the negative or positive weights input. This 'solution blending', depending the inputs signs, is trying to solve the common gradient vanishing and 'dying ReLU' problems simultaneously and it shows significance positive impact on training and classification accuracy of DNNs as it is concluded from the results of the numerical evaluation performed. The combination of AFs to increase NN performance appears to have remained unknown in the literature review, with an exception of some the recent works (Renlong Jie et al., 2020), (Franco Manessi et al., 2019). The outline of this paper is as follows: Section 2 contains one of the main contributions of this work, which includes the implementation of SigmoReLU in Keras. Section 3 presents experimental results of the proposed AF, including an evaluation of the accuracy of the training. Also it is compared to other well defined AFs in the field. Finally, discussion and the main conclusions of the work are devoted on Section 4.

2. METHODS AND ALGORITHMS

2.1 Datasets and NN models hyperparameters

The following data sets for image, text and tabular data classification that were also used on ALReLU paper (Mastromichalakis, 2020) are employed in the experiments described and discussed in this study:

- COVID-19 X-Rays and Pneumonia Datasets (Kaggle) having 372 and 5,863 X-Ray images respectively.(https://www.kaggle.com/bachrr/covid-chest-xray?select=images, https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia)
- Parkinson Detection Spiral Drawing & Waves (Kaggle) having 204 total images of Waves and Spirals draws images (https://www.kaggle.com/kmader/parkinsons-drawings)
- Histopathologic Cancer Detection Dataset having 57,458 32x32 colour images and it was used in Kaggle competition on 2019 (https://www.kaggle.com/c/histopathologic-cancer-detection/data)
- Quora Insincere Questions Detection Dataset consist by 1,681,928 unique questions in total labeled with 0 if considered not toxic and 1 if considered toxic. It was also used in Kaggle competition on 2019 (https://www.kaggle.com/c/quora-insincere-questions-classification/data)
- Microsoft Malware Prediction was used in Kaggle competition on 2019 too, and has 16,774,736 records in tabular format (https://www.kaggle.com/c/microsoft-malware-prediction/data)

The CNN-related hyperparameters to train the datasets of COVID-19, Spiral Drawing & Waves and Histopathologic Cancer Detection data set is a deep CNN and has the following layers:

- five convolutional layers, the first of which has kernel size of 5x5 and the four last 3x3
- the following convolutional filters for each of the two convolutional layers respectively (in order from the first layer to the last one): 32, 64, 128, 256, 512
- Max Pooling and Batch Normalization is applied after all convolutional layers;

- Dropout layer also after all convolutional layers leveraged with dropout rates (in order from the first layer to the last one): 0.1, 0.2, 0.3, 0.4, 0.5
- The AFs were also applied after all convolution layers
- A Global Average Pooling Layer followed by AF, Batch Normalization and dropout with rate 0.3
- A Dense layer of size 256 followed by AF, Batch Normalization and dropout with rate 0.4
- a final Dense layer with softmax activation, having as size of neurons the number of output classes of each the dataset

On Quora Insincere Questions Dataset, it was used a Bidirectional LSTM CNN, set are as follows:

- As embedding layer, it was used the mean of both Glove and Paragram embendings (these embeddings are included in the Quora Kaggle Dataset that is referred before.)
- A Spatial Dropout at 0.2 rate
- A Bidirectional LSTM layer with 128 RNN units,
- four convolutional layers, each of which has a kernel size 1, 2, 3, 5 and filters of 100, 90, 30, 12 respectively
- Each convolution layers is followed by the AF
- four Global Max Pooling layer
- A Dense layer of 200, followed by AF dropout (0.2) and batch normalization
- a final Dense layer with softmax activation, having 2 neurons as for each class (toxic, not toxic)

A shallow neural network model is used to train Microsoft Malware Prediction dataset with the following properties:

- A Dense layer of size 100, followed by dropout rate of 0.4, Batch Normalization and AF
- Another Dense layer of size 100 , followed by dropout rate of 0.4, Batch Normalization and AF
- a final Dense layer with softmax activation, having 2 neurons as for each class (has malware detection, not has malware detection)

2.2 The SigmoReLU AF

Rectified Linear Unit, or ReLU, is commonly used on between layers to add nonlinearity in order to handle more complex and nonlinear datasets. Fig. 1, demonstrates the ReLU that can be expressed as follows (Eq. (2) is ReLU derivative):

$$f(x) = \begin{cases} 0 \ \forall \ x < 0 \\ x \ \forall \ x > 0 \end{cases} \qquad (1)$$

$$\frac{dy}{dx} f(x) = \begin{cases} 0 \ \forall \ x < 0 \\ 1 \ \forall \ x > 0 \end{cases} \qquad (2)$$

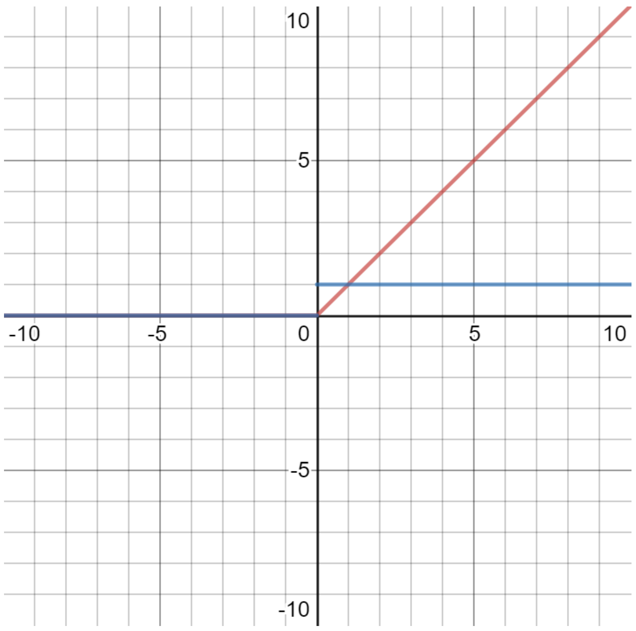

Figure 1: Red: ReLU AF, Blue: ReLU Derivative

The issues of ReLU come with the fact that it is not differentiable x=0, since ReLU sets all values < 0 to zero. Although it can benefit on sparse data, when gradient is 0 the neurons arriving at large negative values and cannot recover from being stuck at 0. The neuron at this stage effectively dies. This is known as the 'dying ReLU' problem and can leads the network essentially stops learning and underperforms.

Current improvements to the ReLU, such as the LReLU, allow for a more non-linear output to either account for small negative values or facilitate the transition from positive to small negative values, without eliminating the problem though.

Sigmoid on the other hand, suffers of Vanishing Gradient problem. If the Sigmoid weights input (in absolute value) is too large, the gradient of the sigmoid function becomes too small. The disadvantage of this is that if you have many layers (i.e. DNN), you will multiply these gradients, and the product of many smaller than 1 values goes to zero very quickly. For this reason, Sigmoid and its relatives such as tanh, are not suitable for training DNN. This problem has been solved by ReLU, which the gradient is either 0 for x<0 or 1 for x>0 meaning that you can put as many layers as you like, because multiplying the gradients will neither vanish nor explode. This is the reason that ReLU is more commonly used in DNNs the last years. In Eq. (3), Eq. (4) and Fig.2 the Sigmoid and its derivative are demonstrated.

$$f(x) = \left\{ \frac{1}{e^{-x}+1} \right. \tag{3}$$

$$\frac{dy}{dx} f(x) = \left\{ \frac{e^{x}}{(e^{x}+1)^{2}} \right. \tag{4}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 9 June 2021 doi:10.20944/preprints202106.0252.v1

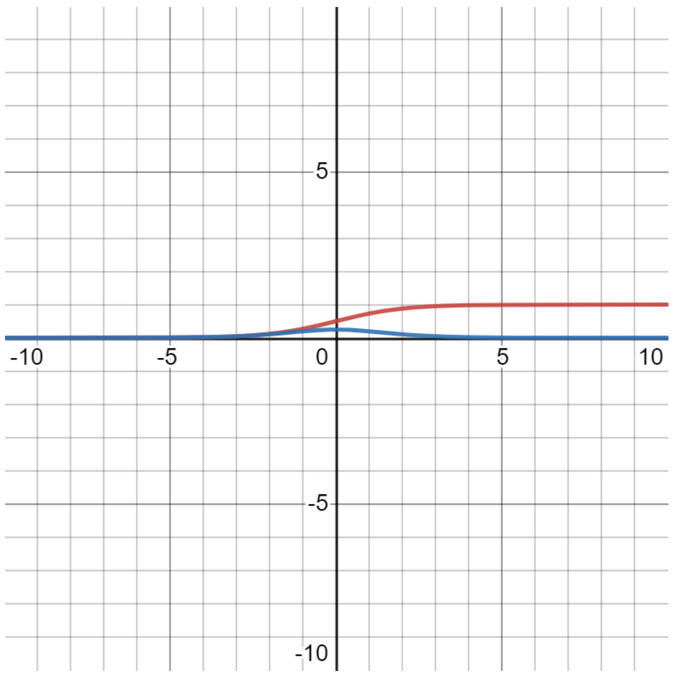

Figure 2: Red: Sigmoid AF, Blue: Sigmoid Derivative

Although ReLU is commonly more robust and useful than Sigmoid, the later has also some advantages in different situations. Actually, there are times that Sigmoid can perform better that ReLU. Consequently, this study investigates the development of a new AF that take both the best advantages of these two functions, to obviate the problem of the 'dying ReLU' and the vanishing gradient of Sigmoid. This is achieved by using the ReLU gradient solution if of x>0 or Sigmoid(x) < x and Sigmoid's gradient solution on exp(-x) > -1 or Sigmoid(x) ≥ x (Eq. 6). Although it is obvious that derivative of this new function is not differentiable everywhere as it demonstrated in Fig. 3 and Eq. (6), it seems that is not cause serious problems, and not impact training performance. Instead, the experiments and results on Section 3, indicate a significance positive impact on performance of this functions combination.

$$f(x) = \begin{cases} \mathrm{relu(x)} \ \forall \ \mathrm{max(0, x)} > \dfrac{1}{e^{-x}+1} \\ sigmoid(x) \ \forall \ \dfrac{1}{e^{-x}+1} > \mathrm{max(0, x)} \end{cases} \tag{5}$$

$$\frac{dy}{dx} f(x) = \begin{cases} 1 \ \forall \ \mathrm{x} > 0 \wedge \dfrac{1}{e^{-x}+1} < x \\ \dfrac{e^{x}}{(e^{x}+1)^{2}} \ \forall \ e^{-x} > \text{-}1 \wedge \dfrac{1}{e^{-x}+1} \geq x \end{cases} \tag{6}$$

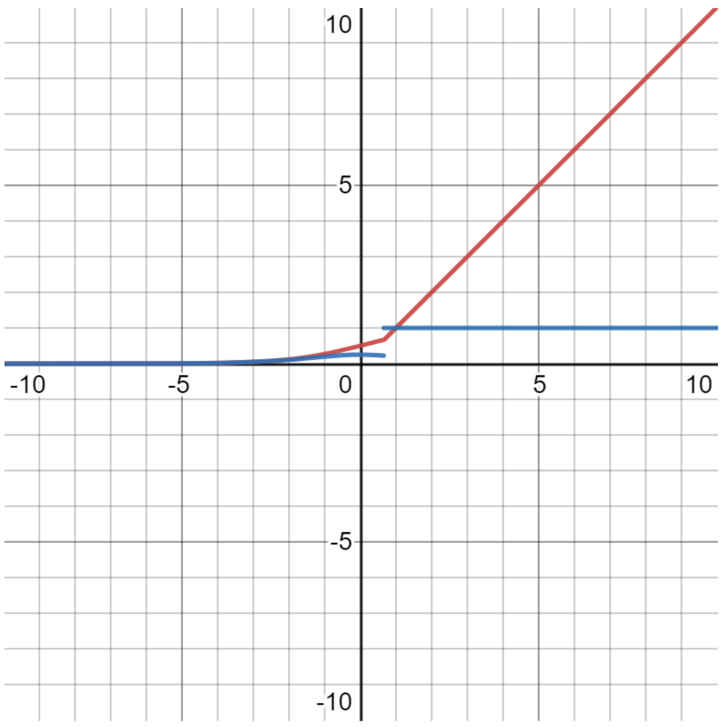

Figure 3: Red: SigmoReLU AF, Blue: SigmoReLU Derivative

The following code snippets, show the Keras implementation of the proposed AF, as well as its usage after the Convolution Layers. The derivatives and gradients of AFs are automatically calculated in TensorFlow (TF) 2, so they have not implemented manually.

Listing 1: A snippet of code in Python (Keras) with SigmoReLU implementation and usage

```
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Conv2D, Lambda
from tensorflow.keras.utils import get_custom_objects
def SigmoReLU(x):
   return K.maximum(tf.keras.actications.relu(x),
tf.keras.actications.sigmoid(x))

get_custom_objects().update({'SigmoReLU':
tf.keras.layers.Activation(SigmoReLU)})

conv = Conv2D(32, (5, 5))(visible)
conv_act = SigmoReLU(conv)
conv_act_batch = BatchNormalization()(conv_act)
conv_maxpool = MaxPooling2D()(conv_act_batch)
conv_dropout = Dropout(0.1)(conv_maxpool)
```

Listing 2: A snippet of code in Python (Keras) with SigmoReLU usage in Bidirectional LSTM CNN

```
x1 = Conv1D(filters=100, kernel_size=1,
                padding='same')(x)
x1 = Activation(SigmoReLU)(x1)
x2 = Conv1D(filters=90, kernel_size=2,
            padding='same')(x)
x2 = Activation(SigmoReLU)(x2)
x3 = Conv1D(filters=30, kernel_size=3,
            padding='same',)(x)
x3 = Activation(SigmoReLU)(x3)
x4 = Conv1D(filters=12, kernel_size=5,
            padding='same')(x)
x4 = Activation(SigmoReLU)(x4)

x1 = GlobalMaxPool1D()(x1)
x2 = GlobalMaxPool1D()(x2)
x3 = GlobalMaxPool1D()(x3)
x4 = GlobalMaxPool1D()(x4)

c = Concatenate()([x1, x2, x3, x4])
x = Dense(200)(c)
x = Activation(SigmoReLU)(x)
x = Dropout(0.2)(x)
x = BatchNormalization()(x)
x = Dense(2, activation="softmax")(x)
```

3. EXPERIMENTAL RESULTS

In order to estimate the performance of training and classification accuracy a 5-Fold validation used in every dataset. The 5-Fold validation procedure has been executed 10 times for every model and dataset to handle the uncertainty caused by GPU and TF. The average results are demonstrated in this section and it's obvious that support the theoretical superiority of the proposed SigmoidReLU AF when compared to well-established ReLU and LReLU AFs. The classification performance results are demonstrated in Table 1 and are described above:

	Performance measures	SigmoReLU (this study)	ReLU	Leaky ReLU
COVID-19 X-Rays (Kaggle)	**Weighted Precision**	80%	46%	45%
	Accuracy	77.42%	62.9%	59.68%
	Weighted Recall	77%	63%	60%
	AUC	89.137%	91.51%	86.63%
	Weighted F1	76%	52%	49%
Spiral Drawings+Waves (Kaggle)	**Weighted Precision**	73%	76%	79%
	Accuracy	68.18%	54.45%	63.64%
	Weighted Recall	68%	55%	64%
	AUC	80.99%	70.24%	71.07%
	Weighted F1	66%	43%	58%
Histopathologic Cancer Detection (Kaggle)	**Weighted Precision**	88%	87%	87%
	Accuracy	88.06%	85.65%	85.32%
	Weighted Recall	88%	86%	85%
	AUC	94.93%	94.65%	95.1%
	Weighted F1	88%	85%	85%
Quora Insincere Questions (Kaggle)	**Weighted Precision**	96%	96%	96%
	Accuracy	96.35%	96.36%	96.36%
	Weighted Recall	96%	96%	96%
	AUC	97.29%	97.30%	97.29%
	Weighted F1	96%	96%	96%
Microsoft Malware Prediction (Kaggle)	**Weighted Precision**	65%	65%	65%
	Accuracy	64.57%	64.72%	64.74%
	Weighted Recall	65%	65%	65%
	AUC	70.73%	71%	70.88%
	Weighted F1	65%	65%	65%

Table 1: Classification performance measures for SigmoReLU, ReLU and LReLU, various datasets

- ***CNN train and classification on COVID-19 X-Rays***: All the evaluation metrics, except AUC are better than original ReLU and LReLU activation functions. The most important is that the accuracy is 14% and 17% better that ReLU and LReLU respectively. This demonstrated in the Table 1 first row dataset "COVID-19 X-Rays (Kaggle)".

- ***CNN train and classification on Spiral Drawings and Waves for Parkinson disease prediction***: The results show a better performance of the proposed AF on Weighted Accuracy, Weighted Recall, F1, and AUC. Only the Precision was a little lower (73% over 76% and 79%) comparing with original ReLU and LReLU. The accuracy here is also significant improvement over ReLU (>13.7300) and LReLU (>4.54) (Table 1, second row "Spiral Drawings+Waves (Kaggle)".

- ***For CNN train and classification of Histopathologic Cancer Detection Dataset***: All the evaluation metrics of SigmoReLU are significant improvement over other AFs. (Table 1, third row "Histopathologic Cancer Detection (Kaggle)").

- ***For LSTM CNN train and classification of Quora Insincere Questions:*** The 3 AF have almost similar performance in this case. (Table 1, third row "Quora Insincere Questions (Kaggle)").

- ***For NN train and classification of Microsoft Malware Prediction Dataset*****:** In this tabular The 3 AF have almost similar performance in this case. (Table 1, third row "Microsoft Malware Prediction (Kaggle)").

4. CONCLUSION

In this paper, a new AF was proposed by employing the combination of two different AFs i.e ReLU and Sigmoid. The technique was found to be highly effective and more accurate for training and classification procedures in DNNs. The proposed AF is merely centering on the improvement of the classification accuracy in sections wherein high accuracy and reliability is very important to be achieved, such as image (including tomography), text and tabular data classification. Numerous tests and measures were performed in order to validate the theoretical framework developed in this paper. The proposed method is used in several experiments of training and classification and it shows superiority to the state-of-the-art ReLU and LReLU in terms of ROC/AUC metrics, recall, precision, F1 scores and accuracy, especially on image training on CNN. On the other hand, on LSTM and shallow NN for tabular data the results are almost similar to the compared functions. The important conclusion and contribution of this work is that the combination of different AFs can also combine the single advantages of them and achieve more accurate and robust results by using both two different AFs solutions, depending the input sign. By this way, they are solved both vanishing gradient and 'dying ReLU' problems at the same time. It is also important that the proposed combination has very high performance in terms of accuracy in COVID-19 image classification. In the future work, different combinations of AFs may proposed and tested, such as ReLU and tanh.

5. REFERENCES

Luca Parisi, Daniel Neagu, Renfei Ma, and Felician Campean. Qrelu and m-qrelu: Two novel quantum activation functions to aid medical diagnostics. arXiv preprint arXiv:2010.08031, 2020a

Luca Parisi, Renfei Ma, Narrendar RaviChandran, Matteo Lanzillotta: hyper-sinh: An Accurate and Reliable Function from Shallow to Deep Learning in TensorFlow and Keras. arXiv preprint arXiv:2011.07661, 2020b

Renlong Jie, Junbin Gao, Andrey Vasnev, Min-ngoc Tran: Regularized Flexible Activation Function Combinations for Deep Neural Networks arXiv preprint arXiv:2007.13101v2, 2020

Franco Manessi, Alessandro Rozza: Learning Combinations of Activation Functions. arXiv preprint arXiv:1801.09403v3, 2019

Stamatis Mastromichalakis: A different approach on Leaky ReLU activation function to improve Neural Networks Performance. arXiv preprint arXiv:2012.07564v2, 2020

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. Proceedings of the 30th International Conference on Machine Learning. Atlanta, Georgia, USA: JMLR: W&CP.

Deep Learning (Ian J. Goodfellow, Yoshua Bengio and Aaron Courville), MIT Press, 2016.