Customization of Activation Layer for Classification

Name:	**Shivam Shukla**
Roll No.:	19301
Institute Name:	IISER Bhopal
Program/Stream:	EECS
Date of Submission:	November 24, 2022

1 Overview

Activation functions are essential in modeling more complicated relationships and patterns in the data using deep learning models. The activation functions that are traditionally used, including Sigmoid, Tanh, and ReLU, can be adaptively customized in this study by merely changing a small number of parameters. Some theoretical and practical analysis on accelerating convergence and enhancing performance is offered. A series of tests are carried out on the weights of activation functions using different network models, i.e., AlexNet, LeNet, DenseNet, GoogleNet, and ResNet, to confirm the effectiveness of the suggested methodology.

2 Research Question

To develop a customized activation function that can enhance the non-linearity of neural networks through traditional and adaptive activation functions and improve their ability to connect the inputs to the response variables.

3 Background and Prior Work

In the artificial neural network community, a wide range of activation functions have been proposed during the past few decades. Activation functions can be categorized into two groups: fixed activation functions and adaptive activation functions, depending on whether a parameter or shape of an activation function is learnable or modifiable during the training phase.

Fixed activation functions suggest that the parameters cannot be adjusted during the training phase. The most frequent fixed activation functions can be divided into three categories: Logistic (Sigmoid), Hyperbolic Tangent (Tanh), and Rectified Linear Activation (as shown in figure 1).

In neural networks, 'adaptive activation functions' are defined as the functions whose parameters are learned and trained together with other parameters, hence adaptively varying with training data. To put it another way, the fundamental goal of this type of function is to find an appropriate function shape leveraging knowledge from training data.

For instance, PReLU [1] substitutes a trainable parameter α_i for the fixed slope α of LReLU [2] in the negative region. Swish [3] activation function that bridges the gap between the linear function and the ReLU function. It has no upper bound, lower bound, smoothness, or non-monotonicity. Other comparable activation mechanisms, such as FReLU [4] and PELU [5], have improved performance in some particular tasks (as shown in figure 2).

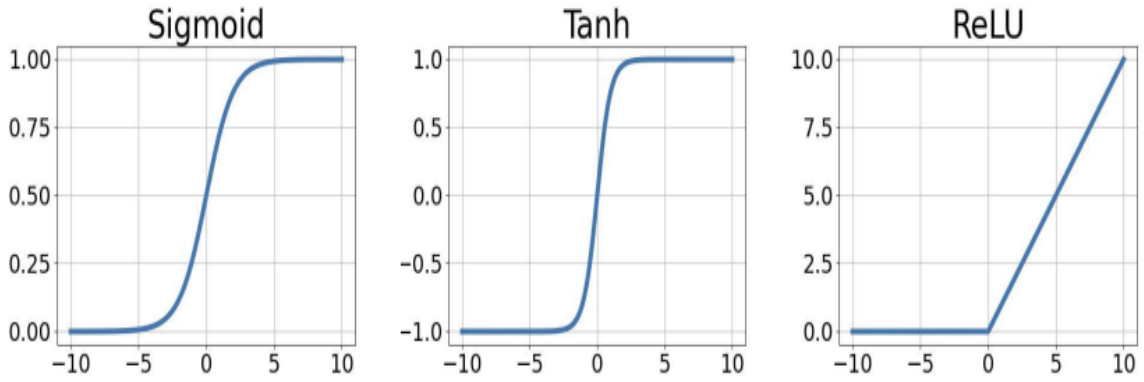

Figure 1: Fixed activation functions with a fixed shape

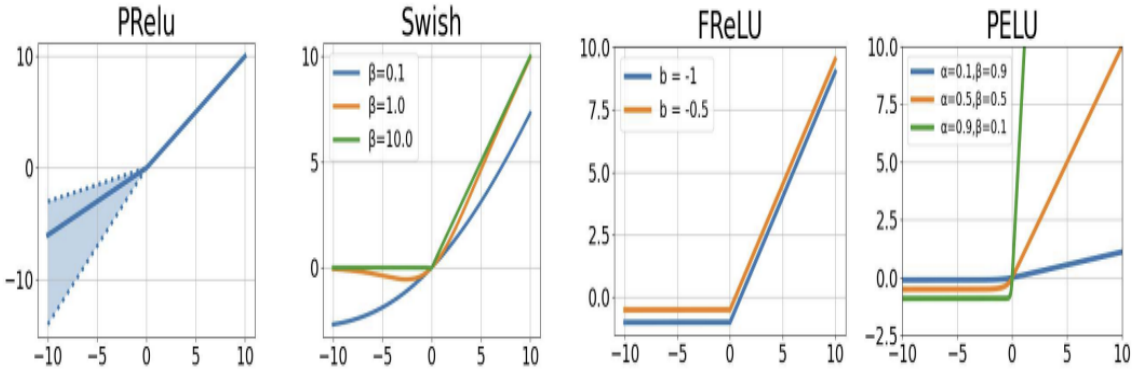

Figure 2: Adaptive activation functions with an adjustable shape

4 Hypothesis

The goal of the adaptive activation function is to achieve rapid convergence by altering the learning rate. To do this, the weight w and the parameter to scale the inputs in any layer are mutually adjusted, which accelerates neural network learning and improves classification accuracy.

Fundamentally, training neural networks is a non-convex optimization problem where the best weight parameters can be sought and discovered using the back-propagation algorithm, allowing the activation function to explore and find the functional subspace. By learning hyper-parameters to adapt the affine transformation's parameters to the network input, adaptive activation functions can boost the adaptability and representational capacity of network models. The customized adaptive functions are designed for each layer[6] where only a few additional parameters are added to the traditional activation functions, where i-th layer activation function is defined as follows:

$$f_A(a_i, b_i, c_i, d_i, z) = b_i f(a_i z + c_i) + d_i$$

where f(.) represents the conventional or fixed activation function.
a_i, b_i, c_i, d_i are learnable parameters of the ith layer parameters in the i-th layer, which can efficiently avoid collapsing into local minimums by adapting to the various tasks based on the complexity of the input data.
The weighted sum of inputs, including the bias term, is represented by $z = wx + b$.

- ASigmoid: $f_A\text{sigmoid} = b_i\text{Sigmoid}(a_i z + c_i) + d_i$
- ATanh: $f_A\tanh = b_i\tanh(a_i z + c_i) + d_i$
- AReLU: $f_A\text{relu} = \text{maximum}(a_i z + c_i, b_i z + d_i)$

a_i and c_i are used to scale inputs; b_i and d_i are used to scale outputs.

The proposed **method 1** is the weighted linear combination of the above-mentioned adaptive activation functions, where the weights X, Y and Z are normalized learnable parameters.

$$f(x) = X.ATanh(x) + Y.ASigmoid(x) + Z.AReLU(x) \tag{1}$$

where

$$\begin{aligned} X &= \frac{\alpha}{\alpha + \beta + \gamma} \\ Y &= \frac{\beta}{\alpha + \beta + \gamma} \\ Z &= \frac{\gamma}{\alpha + \beta + \gamma} \end{aligned} \tag{2}$$

and α, β and γ are learnable parameters.
The proposed **method 2** is the weighted linear combination of the traditional adaptive functions, where the weights are normalized learnable parameters.

$$f(x) = X.Tanh(x) + Y.Sigmoid(x) + Z.ReLU(x) \tag{3}$$

After learning the parameters, we train the neural networks with the learned weights.

5 Datasets

The dataset used for verifying and comparing the effectiveness of the discussed activation functions is CIFAR10. It can be downloaded from the following link: https://www.cs.toronto.edu/ kriz/cifar.html

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

Here are the classes in the dataset:

1. airplane
2. automobile
3. bird
4. cat
5. deer
6. dog
7. frog
8. horse
9. ship
10. truck

Table 1: Precision scores of various models on different activation functions on 150 epochs

	AlexNet	LeNet	GoogleNet	ResNet	DenseNet
Sigmoid	0.8575	0.3284	0.7932	0.4639	0.5463
ASigmoid	0.8545	0.4797	0.8990	0.4424	0.5619
Tanh	0.8265	0.4645	0.8922	0.6545	0.7264
ATanh	0.8533	0.6683	0.8598	0.7008	0.7618
ReLU	0.8717	0.5872	0.9241	0.9732	0.8235
AReLU	0.8744	0.6677	0.8759	0.8073	0.8642
Method 1	0.8553	0.6771	0.8737	0.7751	0.9258
Method 2	0.8579	0.5692	0.9258	0.7540	0.9326

6 Setup

Due to the use of GPU for training the models using a remote server, it was not feasible to do the training in a notebook hence training has been done using the python file "main.py". The entire project code base has been uploaded to Google Drive codeBaseLink also attached with the submission files.

- Code has been written in TensorFlow
- main.py consist of training code and saving it for future reference.
- Various models with different activation functions have been coded in separate .py files.
  - AlexNet_model.py for AlexNet
  - LeNet_model.py for LeNet
  - GoogLeNet_model.py for GoogLeNet
  - ResNet_model.py for ResNet50
  - DenseNet_Cifar10_save.py for DenseNet
- Running the single bash file script located at script "/script/script.sh" will generate all the results by training each of the models for the mentioned different activation functions.
- Analysis has been done in Jupyter-Notebook which is also attached with submission files.

7 Descriptive Analysis

The developed adaptive functions were employed on **CIFAR 10** dataset by training on five models, i.e., AlexNet, LeNet, GoogleNet, ResNet, and DenseNet. The produced results of the precision score on 150 epochs using Stochastic Gradient Descent are shown in Table 1.
A similar analysis of precision score with standard deviations has been employed on **AlexNet**, as shown in Table 2. Each configuration of AlexNet for the different activation function have been trained 3 times, and mean maximum precision with standard deviation (error) have obtained from these three trials of training.

The graphs for various activation functions during the training processes are shown in Figure 3.

The models AlexNet and LeNet were trained on the above-mentioned Method 1 and Method 2 to analyze the values of trainable parameters X, Y, and Z for each layer as shown in Table 3 and 4, respectively.

After training, these parameters (X, Y, and Z) were learned as approximately 0.33 each in both methods. These recorded weights, X, Y, and Z, were kept fixed (0.33 each and 1 each), and then the model AlexNet, was trained again on these fixed weights and the maximum precision scores and the precision scores of the last epoch were recorded, as shown in Table 5. The loss and precision curves of AlexNet and LeNet on Method 1 and Method 2 are also shown in Figure 4

Table 2: Analysis of AlexNet on different activation functions on 80 epochs

Activation Functions	Max Precision(at Epoch)	Last Epoch Precision
Sigmoid	0.8575±0.0267(20)	0.6645±0.0437
ASigmoid	0.8545±0.0048(31)	0.7369±0.0305
Tanh	0.8265±0.0135(64)	0.7465±0.0637
ATanh	0.8463±0.015(49)	0.6962±0.0360
ReLU	0.87±0.0207(62)	0.7089±0.0425
AReLU	0.8744±0.0067(28)	0.7224±0.0395
Method 1	0.8553±0.0096(33)	0.7123±0.0331
Method 2	0.8579±0.0059(37)	0.6742±0.0358

Table 3: Analysis of normalized learnable parameters of AlexNet on 50 epochs

	Parameters	Conv1	Conv2	Conv3	Conv4	Conv5	FC6	FC7
Method 1	X	0.338	0.34	0.35	0.351	0.337	0.337	0.34
	Y	0.324	0.317	0.331	0.332	0.334	0.329	0.277
	Z	0.338	0.342	0.319	0.317	0.329	0.335	0.384
Method 2	X	0.333	0.33	0.335	0.334	0.332	0.352	0.404
	Y	0.33	0.302	0.315	0.32	0.332	0.313	0.231
	Z	0.365	0.368	0.35	0.346	0.337	0.335	0.365

Table 4: Analysis of normalized learnable parameters of LeNet on 50 epochs

	Parameters	Conv1	Conv2	FC3
Method 1	X	0.345	0.337	0.349
	Y	0.332	0.334	0.22
	Z	0.322	0.329	0.431
Method 2	X	0.328	0.334	0.35
	Y	0.332	0.334	0.196
	Z	0.34	0.333	0.454

Table 5: Analysis of AlexNet by fixing the parameters on 80 epochs

	Parameter Values	Max Precision(at Epoch)	Last Epoch Precision
Method 1	1	0.8402±0.0143(39)	0.7122±0.0303
	1/3	0.8447±0.0078(42)	0.7311±0.0219
Method 2	1	0.8614±0.0138(38)	0.7637±0.0558
	1/3	0.8697±0.0216(44)	0.6953±0.0360

8 Evaluation Metric

Precision has been used as the evaluation metric in the paper. A classification model's ability to extract only the correctly classified data points is precision. It is computed by dividing the number number of true positives by the total number of true positives and false positives.

9 Conclusion and Discussion

The Method 2 (Linear Combination of Traditional Activation functions) performs best among the discussed activation functions on DenseNet and GoogLeNet architectures, and Method 1 (Linear Combination of Adaptive Activation functions) performs best on the LeNet model in terms of precision. ReLU and AReLU have performed the best on AlexNet and ResNet but the proposed have reasonable results on them. In the weighted linear combination with fixed and adaptive weights (X, Y and Z) for proposed methods, it was observed that activation functions with fixed weights performed better than adaptive weights, and between fixed weights of 0.33 each and 1 each, weight 0.33 performed better than the weight 1.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian 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.

[2] Andrew L Maas, Awni Y Hannun, Andrew Y Ng, et al. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, volume 30, page 3. Atlanta, Georgia, USA, 2013.

[3] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[4] Suo Qiu, Xiangmin Xu, and Bolun Cai. Frelu: flexible rectified linear units for improving convolutional neural networks. In *2018 24th international conference on pattern recognition (icpr)*, pages 1223–1228. IEEE, 2018.

[5] Ludovic Trottier, Philippe Giguere, and Brahim Chaib-Draa. Parametric exponential linear unit for deep convolutional neural networks. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 207–214. IEEE, 2017.

[6] Haigen Hu, Aizhu Liu, Qiu Guan, Hanwang Qian, Xiaoxin Li, Shengyong Chen, and Qianwei Zhou. Adaptively customizing activation functions for various layers. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

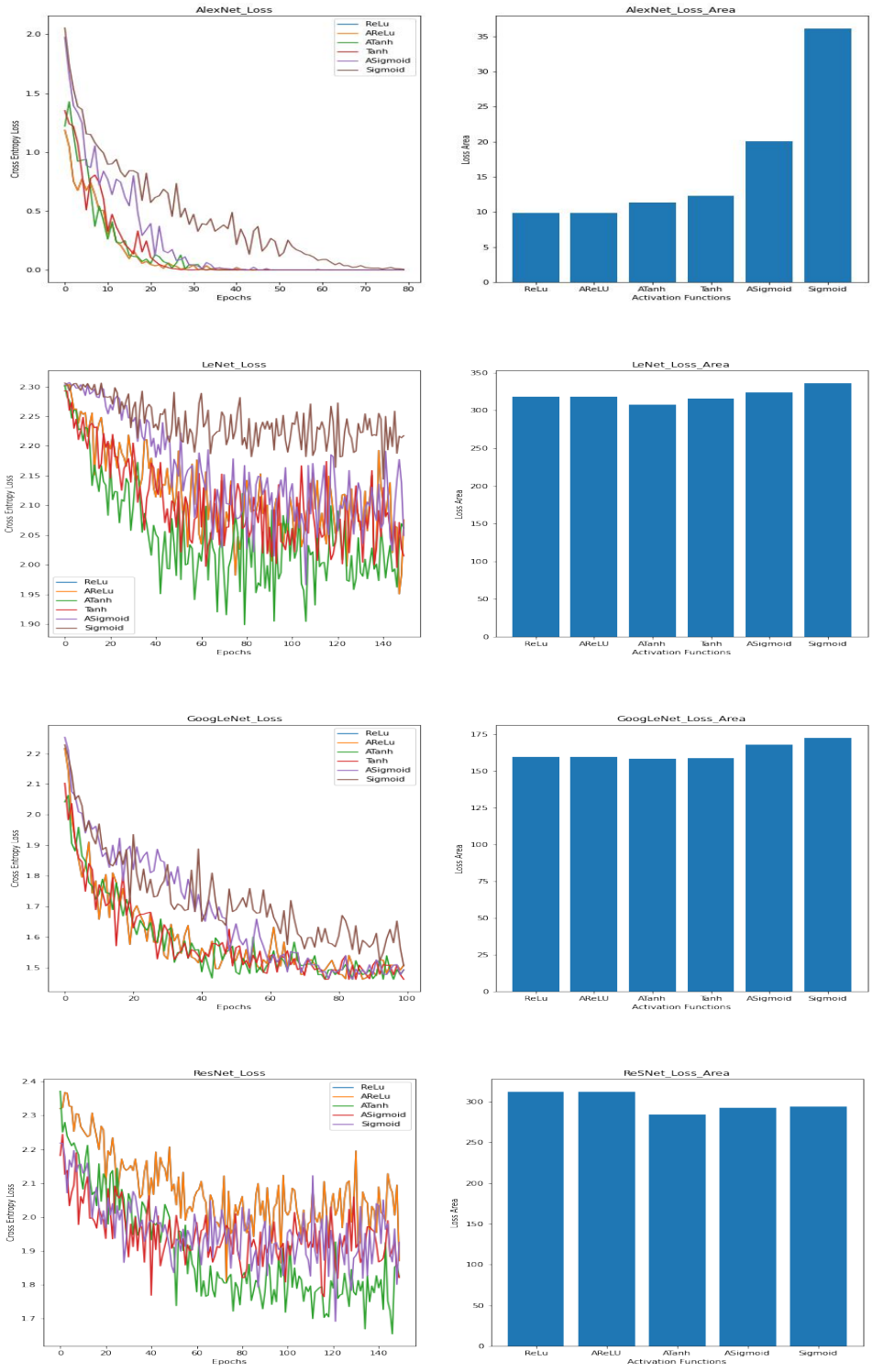

Figure 3: The left graphs represent the loss-epoch convergence curves. The right graphs represent the area enclosed by corresponding convergence curves.

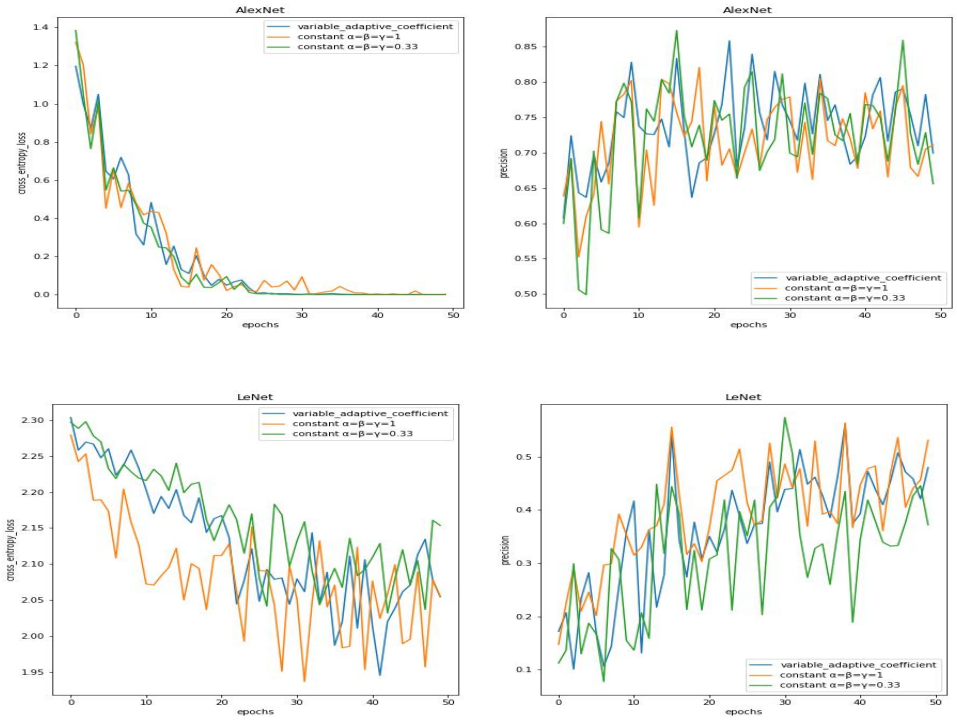

Figure 4: The left graphs represent the loss-epoch convergence curves. The right graphs represent the precision corresponding to epochs.