A Study on Smooth Activation Functions

A Thesis

Submitted in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

by

Koushik Biswas

Roll No: PhD16007

Under the supervision of

Dr. Ashish Kumar Pandey

Dr. Shilpak Banerjee

Department of Computer Science and Engineering

Indraprastha Institute of Information Technology, Delhi

New Delhi- 110020

THESIS CERTIFICATE

This is to certify that the thesis titled **"A Study on Smooth Activation Functions"**, submitted by **Mr. Koushik Biswas**, to the Indraprastha Institute of Information Technology Delhi, for the award of the degree of **Doctor of Philosophy**, is a bonafide record of the research work done by him under my supervision. In my opinion, the thesis has reached the standards fulfilling the requirements of the regulations relating to the degree. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Thesis Supervisors

August, 2023
Dr. Ashish Kumar Pandey
Assistant Professor
Department of Mathematics
IIIT Delhi.

August, 2023
Dr. Shilpak Banerjee
Assistant Professor
Department of Mathematics & Statistics
IIT Tirupati.

ACKNOWLEDGEMENTS

I am really fortunate to get support from several fantastic people during my Ph.D. time. I would like to begin by expressing my sincere gratitude to my advisors, Dr. Ashish Kumar Pandey and Dr. Shilpak Banerjee for their continuous support, guidance, and trust throughout my Ph.D. I would like to thank them for providing me with all the opportunities they gave me to work on the specific thesis topic.

I was really fortunate to be a part of the Cryptology Research Group (CRG), IIIT Delhi, where I was advised by Dr. Somitra Sanadhya and Dr. Donghoon Chang. I would also like to thank Dr. Sourav Mukhopadhyay and Dr. Ratna Dutta for giving me proper guidance during my summer internship at the Indian Institute of Technology, Kharagpur. I am also thankful to Dr. Bapi Chatterjee for providing the necessary advice and computational resources to complete some essential experiments.

Next, I am extremely fortunate to get support from my labmate, Dr. Sandeep Kumar, on every occasion. He helped me professionally and personally whenever I needed any support. I also like to thank my friends Omkar, Mohit, Prawendra, Ridam, Sayantan, and my seniors Dr. Rahul Gangopadhyay and Dr. Amit Kumar Chauhan for their constant support and help throughout my Ph.D.

Finally, I would like to heartily thank my family members and my parents for their constant encouragement and unconditional support in every event of my life.

Koushik Biswas.

PUBLICATIONS

Publications Related to the Dissertation

1. **Koushik Biswas**, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. Smooth Maximum Unit: Smooth Activation Function for Deep Networks using Smoothing Maximum Technique. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2022. Biswas *et al.* (2022)

2. **Koushik Biswas**, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. SAU: Smooth activation function using convolution with approximate identities. In European Conference on Computer Vision (ECCV), 2022. Biswas *et al.* (2021*d*)

3. **Koushik Biswas**, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. ErfAct and Pserf: Non-monotonic smooth trainable Activation Functions. In Proceedings of the AAAI Conference on Artificial Intelligence, 2022. Biswas *et al.* (2021*c*)

4. **Koushik Biswas**, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. TanhSoft—Dynamic Trainable Activation Functions for Faster Learning and Better Performance, in IEEE Access, vol. 9, pp. 120613-120623, 2021, doi: 10.1109/ACCESS.2021.3105355. Biswas *et al.* (2021*e*)

5. **Koushik Biswas**, Sandeep Kumar, Shilpak Banerjee, and Ashish Kumar Pandey. EIS - Efficient and trainable activation functions for better accuracy and performance. In Artificial Neural Networks and Machine Learning –ICANN 2021. Springer International Publishing, Cham, 2021. ISBN 978-3-030-86340-1. Biswas *et al.* (2021*b*)

Preprints Related to the Dissertation

1. **Koushik Biswas**, Shilpak Banerjee, and Ashish Kumar Pandey. Orthogonal-Padé Activation Functions: Trainable Activation functions for smooth and faster convergence in deep networks.

2. **Koushik Biswas**. Maximum Activation Unit: Smooth Activation from Approximation of the Maximum Function.

Other Publications

1. Sandeep Kumar, **Koushik Biswas**, and Ashish Kumar Pandey. Prediction of landfall intensity, location, and time of a tropical cyclone. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 35. 2021. Kumar *et al.* (2021*b*)

2. Sandeep Kumar, **Koushik Biswas**, and Ashish Kumar Pandey. Predicting landfall's location and time of a tropical cyclone using reanalysis data. In Artificial Neural Networks and Machine Learning – ICANN 2021. Springer International Publishing, 2021. ISBN 978-3-030-86380-7. Kumar *et al.* (2021*a*)

3. Sandeep Kumar, **Koushik Biswas**, and Ashish Kumar Pandey. Track prediction of tropical cyclones using long short-term memory network. In 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) 2021. Kumar *et al.* (2021*c*)

4. Sandeep Kumar, **Koushik Biswas**, and Ashish Kumar Pandey. Will a Tropical Cylone make Landfall? Neural Computing and Applications, 2022. Kumar *et al.* (2022*b*)

5. Sandeep Kumar, **Koushik Biswas**, and Ashish Kumar Pandey. Forecasting formation of a Tropical Cyclone Using Reanalysis Data (Communicated). Kumar *et al.* (2022*a*)

ABSTRACT

Artificial neural networks (ANNs) have occupied the centre stage in deep learning. An activation function is a crucial component in the neural network, which introduces the non-linearity in the network. An activation function is considered good if it can generalise better on a variety of datasets, ensure faster convergence and improve neural network performance. The Rectified Linear Unit (ReLU) has emerged as the most popular activation function due to its simplicity though it has some drawbacks. To overcome the shortcomings of ReLU (non-smooth, non-zero mean, negative missing, to name a few), and to increase the accuracy considerably in a variety of tasks, many new activation functions have been proposed over the years like Leaky ReLU, ELU, Softplus, Parametric ReLU, ReLU6 etc. However, all of them provides marginal improvement over ReLU. Swish, GELU, Padé activation unit (PAU), and Mish are some non-linear smooth activations proposed recently which show good improvement over ReLU in a variety of deep learning tasks.

ReLU or its variants are non-smooth (continuous but not differentiable) at the origin though smoothness is an important property during backpropagation. We construct several smooth activation functions, which are approximation by a smooth function of ReLU, Leaky ReLU or its variants. Some of these functions are hand-engineered, while some come from underline mathematical theory. All these functions have shown good improvement over ReLU or Swish in the variety of standard datasets in different deep learning problems like image classification, object detection, semantic segmentation, and machine translation.

KEYWORDS: Smooth Activation Function ; Artificial Neural Network ; Deep Learning

TABLE OF CONTENTS

ACKNOWLEDGEMENTS i

PUBLICATIONS ii

ABSTRACT iv

LIST OF TABLES xix

LIST OF FIGURES xxii

ABBREVIATIONS xxiii

1 INTRODUCTION 1

1.1 Related Works 2

1.1.1 Types of Activation Functions: 3

1.2 Research Contributions 6

2 Proposed Problems and Methodology 9

2.1 Proposed problems 9

2.2 Methodology: 10

3 Smooth Activation Unit 11

3.1 Introduction 11

3.2 Related works and Motivation 11

3.3 Research Contribution 12

3.4 Mathematical formalism 12

3.4.1 Convolution 12

3.4.2 Mollifier and Approximate identities 13

3.4.3 Smooth approximations of non-differentiable functions . . . 14

3.5 Smooth Activation Unit (SAU) 14

3.5.1 Learning activation parameters via back-propagation 15

3.6 Experiments . . . 17
3.6.1 Image Classification . . . 17
3.6.2 Object Detection . . . 27
3.6.3 Semantic Segmentation . . . 28
3.6.4 Machine Translation . . . 28
3.7 Baseline Table . . . 29
3.8 Computational Time Comparison . . . 30
3.9 Conclusion . . . 30

4 Smooth Maximum Unit **32**
4.1 Introduction . . . 32
4.2 Related Works and Motivation . . . 33
4.3 Research contribution . . . 33
4.4 Smooth Maximum Unit . . . 34
4.4.1 Approximation by a smooth function of the maximum function 34
4.4.2 Learning activation parameters via back-propagation . . . 38
4.5 Experiments . . . 39
4.5.1 Image Classification . . . 40
4.5.2 Object Detection . . . 51
4.5.3 Semantic Segmentation . . . 52
4.5.4 Machine Translation . . . 52
4.6 Baseline Table . . . 53
4.7 Computational Time Comparison . . . 53
4.8 Conclusion . . . 55

5 ErfAct and Pserf **56**
5.1 Introduction . . . 56
5.2 Related Works and Motivation . . . 57
5.3 Research contribution . . . 57
5.4 ErfAct and Pserf . . . 58
5.5 Experiments . . . 60
5.5.1 Image Classification . . . 60
5.5.2 Semantic Segmentation . . . 69

5.5.3 Object Detection 69
5.5.4 Machine Translation 71
5.6 Baseline Table 71
5.7 Computational Time Comparison 72
5.8 Conclusion 73

6 Maximum Activation Unit **74**
6.1 Introduction 74
6.2 Related works and Motivation 74
6.3 Research Contribution 75
6.4 Maximum Activation Unit 75
6.4.1 Learning activation parameters via back-propagation 78
6.5 Experiments 79
6.5.1 Image Classification 79
6.5.2 Object Detection 87
6.5.3 Semantic Segmentation 87
6.5.4 Machine Translation 88
6.6 Baseline Table 90
6.7 Computational Time Comparison 90
6.8 Conclusion 91

7 Orthogonal-Padé Activation Unit **92**
7.1 Introduction 92
7.2 Related works and motivation 92
7.3 Research contribution 93
7.4 Padé activation Unit (PAU) and Orthogonal-PAU 93
7.4.1 Padé activation Unit (PAU) 94
7.4.2 Orthogonal-Padé activation Unit (OPAU) 95
7.4.3 Learning activation parameters via back-propagation 96
7.5 Networks with orthogonal-Padé activations and function approximation 96
7.6 Appximation coefficients for different orthogonal polynomials . . . 97
7.7 Experimental results with Orthogonal-Padé Activation 99
7.7.1 Image Classification 100

7.7.2 Object Detection . . . 107
7.7.3 Semantic Segmentation . . . 108
7.7.4 Machine Translation . . . 108
7.8 Comparison With the baseline activation functions . . . 109
7.9 Computational Time Comparison . . . 110
7.10 Conclusion . . . 111

8 Tanhsoft **112**
8.1 Introduction . . . 112
8.2 Related works and Motivation . . . 113
8.3 Research contribution . . . 114
8.4 TanhSoft-1, TanhSoft-2, and TanhSoft-3 & their properties . . . 114
8.5 Experiments with TanhSoft-1, TanhSoft-2, and TanhSoft-3 . . . 117
8.5.1 Image Classification: . . . 118
8.5.2 Object Detection . . . 123
8.5.3 Semantic Segmentation . . . 125
8.5.4 Machine Translation . . . 125
8.6 Comparison With Baselines . . . 126
8.7 Computational Time Comparison . . . 127
8.8 Conclusion . . . 127

9 EIS **129**
9.1 Introduction . . . 129
9.2 Related works . . . 130
9.3 Research Contribution . . . 130
9.4 EIS-1, EIS-2, and EIS-3 . . . 131
9.5 Experiments with EIS-1, EIS-2, and EIS-3 . . . 133
9.5.1 Image Classification: . . . 134
9.5.2 Object Detection . . . 137
9.5.3 Semantic Segmentation . . . 138
9.5.4 Machine Translation . . . 138
9.6 Computational Time Comparison . . . 139

9.7 Conclusion 140

10 Conclusion and Future Research Directions **141**

10.1 Conclusion: 141

10.2 Summary of the proposed works 142

10.2.1 Mathematical Summary of existing and proposed works . . 142

10.2.2 Experimental Summary of the proposed works 143

10.3 Future Direction 145

LIST OF TABLES

3.1 A Detailed Comparison between SAU Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with LeNet Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 18

3.2 A Detailed Comparison between SAU Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with AlexNet Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 18

3.3 A Detailed Comparison between SAU Activation and Other Baseline Activations On MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with VGG16 Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 19

3.4 A Detailed Comparison Between SAU Activation and Other Baseline Activations in MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem on Custom-designated Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 19

3.5 A Detailed Comparison between SAU Activation and Other Baseline Activations on The CIFAR100 Dataset for Image Classification Problem with Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 21

3.6 A Detailed Comparison between SAU Activation and Other Baseline Activations on The CIFAR10 Dataset for Image Classification Problem with Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 22

3.7 Experimental Results for Baseline Activations in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.6 given in the CIFAR Section. 22

3.8 Experimental Results for Baseline Activations and SAU in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 23

3.9 Experimental Results for Baseline Activations in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.6 given in the CIFAR Section. 23

3.10 Experimental Results for Baseline Activations and SAU in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. 24

3.11 Experimental Results for Baseline Activations in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.5 given in the CIFAR Section. 24

3.12 Experimental Results for Baseline Activations in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.5 given in the CIFAR Section. 25

3.13 Top-1 Test Accuracy Reported with Mixup Augmentation Method on CIFAR100 Dataset for the Mean of 10 Different Runs. mean±std is Reported in the Table 25

3.14 A Detailed Comparison between SAU Activation and Other Baseline Activations in Tiny ImageNet Dataset for Image Classification Problem. We Report top-1 Test Accuracy (in %) for the Mean of 6 Different Runs. mean±std is Reported in the Table. 26

3.15 top-1 Accuracy Reported on ImageNet-1k Dataset. 27

3.16 A Detailed Comparison between SAU Activation and Other Baseline Activations on Pascal VOC Dataset for Object Detection Problem with SSD300 Network Architecture. We Report mAP for the Mean of 6 Different Runs. mean±std is Reported in the Table. 27

3.17 A Detailed Comparison between SAU Activation and Other Baseline Activations in CityScapes Dataset for Semantic Segmentation Problem on U-NET Model. We Report Pixel Accuracy and mIOU for the Mean of 6 Different Runs. mean±std is Reported in the Table. 28

3.18 A Detailed Comparison between SAU Activation and Other Baseline Activations in WMT-2014 Dataset for Machine Translation Problem on Transformer Model. We Report BLEU Score for the Mean of 6 Different Runs. mean±std is Reported in the Table. 29

3.19 Baseline Table for SAU. In the Table, We Report the Total Number of Cases in Which SAU Underperforms, Equal, or Outperforms When We Compare with the Baseline Activation Functions 29

3.20 Runtime comparison for the forward and backward passes for SAU and other baseline activation functions for a 32× 32 RGB image in ResNet-18 model. 30

4.1 Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on VGG16 architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 41

4.2 Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on LeNet architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 41

4.3 Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on AlexNet architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 42

4.4 Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on custom designed architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 42

4.5 Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 44

4.6 Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR10 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 45

4.7 Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 46

4.8 This is an extension to the Table-4.5 (4.5). We report Top-1 test accuracy (in %) on CIFAR100 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. SF V2 stands for ShuffleNet v2. 47

4.9 This is an extension to the Table-4.5 (4.5). We report Top-1 test accuracy (in %) on CIFAR100 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. 47

4.10 This is an extension to the Table-4.6 (4.6). We report Top-1 test accuracy (in %) on CIFAR10 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. 48

4.11 Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR10 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. 48

4.12 This is an extension to the Table-4.6 (4.5). We report Top-1 test accuracy (in %) on CIFAR10 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. SF V2 stands for ShuffleNet v2. 49

4.13 Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem with Mixup augmentation method. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table. . . 49

4.14 Comparison between SMU, SMU-1 activations and other baseline activations on Tiny ImageNet dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table. 50

4.15 Top-1 accuracy reported on ImageNet-1k dataset. 50

4.16 Comparison between SMU, SMU-1 activations and other baseline activations on Pascal VOC dataset for object detection problem. We report mAP for the mean of 10 different runs. mean±std is reported in the table. 51

4.17 Comparison between SMU, SMU-1 activations and other baseline activations on CityScapes dataset for semantic segmentation problem. We report pixel accuracy and mIOU for the mean of 10 different runs. mean±std is reported in the table. 52

4.18 Comparison between SMU, SMU-1 activations and other baseline activations on WMT2014 dataset for machine translation problem. We report BLEU score for the mean of 10 different runs. mean±std is reported in the table. 53

4.19 Baseline table for SMU. These numbers represent the total number of models in which SMU underperform, equal or outperform compared to the baseline activation functions 54

4.20 Runtime comparison for the forward and backward passes for SMU and SMU-1 and other baseline activation functions for a 32× 32 RGB image in ResNet-18 model. 54

5.1 Comparison between different baseline activations and ErfAct and Pserf activations on MNIST, Fashion MNIST, and SVHN datasets in AlexNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 62

5.2 Comparison between different baseline activations, ErfAct, and Pserf activations on MNIST, Fashion MNIST, and SVHN datasets on VGG-16 network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 62

5.3 Comparison between different baseline activations and ErfAct and Pserf on MNIST, Fashion MNIST, and SVHN datasets with Custom designed network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 63

5.4 Comparison between different baseline activations and ErfAct and Pserf on MNIST, Fashion MNIST, and SVHN datasets with LeNet model. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 63

5.5 Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table. 65

5.6 Comparison between different baseline activations and ErfAct and Pserf on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table. 65

5.7 Comparison between different baseline activations and ErfAct and Pserf on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table. 66

5.8 Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table. 66

5.9 Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) with Mixup augmentation method for mean of 12 different runs have been reported. mean±std is reported in the table. 67

5.10 Comparison between different baseline activations and ErfAct and Pserf on Tiny ImageNet dataset. Mean of 5 different runs for Top-1 accuracy(in %) have been reported. mean±std is reported in the table. . . 68

5.11 Top-1 Accuracy reported on ImageNet-1k dataset. 68

5.12 Comparison between different baseline activations and ErfAct and Pserf on semantic segmentation problem on U-NET model in CityScapes dataset. mean±std is reported in the table. 69

5.13 Comparison between different baseline activations and ErfAct and Pserf on Object Detection problem on SSD 300 model in Pascal-VOC dataset. mean±std is reported in the table. 70

5.14 Comparison between different baseline activations and ErfAct and Pserf on Machine translation problem on transformer model in WMT-2014 dataset. mean±std is reported in the table. 71

5.15 Baseline table for ErfAct and Pserf. These numbers represent the total number of models in which ErfAct and Pserf underperforms, equal or outperforms compared to the baseline activation functions 72

5.16 Runtime comparison for the forward and backward passes for ErfAct and Pserf and baseline activation functions for a 32× 32 RGB image in PreActResNet-18 model. 72

6.1 A Detailed Comparison between MAU-1, MAU-2, and MAu-3 and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with AlexNet Model. Top-1 Test Accuracy (in %) is reported for the Mean of 20 Different Runs. mean±std is reported in the Table. 80

6.2 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with VGG16 Model. Top-1 Test Accuracy (in %) for the Mean of 20 Different Runs is reported. mean±std is reported in the table. 80

6.3 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 82

6.4 Experimental Results for MAU-1, MAU-2, MAU-3 and Baseline Activations in CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 82

6.5 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 83

6.6 Experimental Results for MAU-1, MAU-2, MAU-3 and Baseline Activations in CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 83

6.7 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 84

6.8 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 84

6.9 Top-1 Test Accuracy Reported with Mixup Augmentation Method on CIFAR100 Dataset for the Mean of 20 Different Runs. I report mean±std in the Table. 85

6.10 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activation's in Tiny ImageNet Dataset for Image Classification Problem. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table. 86

6.11 top-1 Accuracy Reported on ImageNet-1k Dataset. 87

6.12 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations on Pascal VOC Dataset for Object Detection Problem with SSD300. I Report mAP for the Mean of 12 Different Runs. I report mean±std in the Table. 88

6.13 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations for Semantic Segmentation Problem in CityScapes Dataset with U-NET Model. I Report Pixel Accuracy and mIOU for the Mean of 12 Different Runs. mean±std is Reported in the Table. 89

6.14 A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations in WMT-2014 Dataset for Machine Translation Problem on Transformer Model. I Report BLEU Score for the Mean of 12 Different Runs. mean±std is Reported in the Table. 89

6.15 Baseline Table for MAU-1, MAU-2, & MAU-3. In the Table, I Report the Total Number of Cases in Which the proposed functions Underperforms, Equal, or Outperforms When compared with the Baseline Activation Functions 90

6.16 Runtime comparison for the forward and backward passes for MAU-1, MAU-2, & MAU-3 and baseline activation functions for a 32× 32 RGB image in PreActResNet-18 model. 91

7.1 Some well-known Orthogonal Polynomial Basis. 94

7.2 Coefficient Table for Leaky ReLU rational function approximation with orthogonal basis (using equation (6)) for network initialization. 'PC' stands for polynomial coefficients. 98

7.3 Comparison between different baseline activations, HP-1, and HP-2 activations on MNIST, Fashion MNIST, and SVHN datasets on VGG-16 network. We report results for 10-fold mean accuracy (in %). mean±std is reported in the table. 101

7.4 Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in AlexNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 101

7.5 Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in LeNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 102

7.6 Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in Custom network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table. 102

7.7 Comparison between different baseline activations, HP-1, and HP-2 activations on CIFAR10 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table. 104

7.8 Comparison between different baseline activations and safe OPAU activations on CIFAR10 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table. 105

7.9 Comparison between different baseline activations and safe OPAU activations on CIFAR100 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table. 106

7.10 Comparison between different baseline activations, HP-1, and HP-2 activations on CIFAR100 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table. 107

7.11 Comparison between different baseline activations, HP-1, and HP-2 activations on the Image classification Problem. We report results for mean of 5 different runs on WRN 28-10 network on Tiny Imagenet Dataset. mean±std is reported in the table. 109

7.12 Comparison between different baseline activations, HP-1, and HP-2 activations on the Object Detection Problem. We report results on SSD 300 with VGG-16 backbones on Pascal-VOC dataset. mean±std is reported in the table. 109

7.13 Comparison between different baseline activations, HP-1, and HP-2 activations on semantic segmentation Problem. We report results on U-NET network on the Cityscapes dataset. mean±std is reported in the table. 110

7.14 Comparison between different baseline activations, HP-1, and HP-2 activations on Machine translation Problem. We report results on Multi-head transformer network on the WMT-2014 dataset. mean±std is reported in the table. 110

7.15 Baseline table for HP-1 and HP-2. These numbers represent the total number of networks in which HP-1 and HP-2 outperforms, equal or underperforms when we compare with the baseline activation functions 110

7.16 Runtime comparison for the forward and backward passes for HP-1, HP-2, and baseline activation functions for a 32× 32 RGB image in VGG-16 model. 111

8.1 Experimental results on MNIST dataset. 118

8.2 Experimental results on Fashion MNIST dataset. 119

8.3 Experimental results on SVHN dataset. 120

8.4 Experimental results on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported. 121

8.5 Experimental results on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported. 121

8.6 Experimental results on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported. 122

8.7 Experimental results on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported. 122

8.8 Experimental results on Tiny ImageNet dataset. Mean of 5 different runs for top-1 accuracy(in %) have been reported. 124

8.9 Object Detection results on SSD 300 model in Pascal-VOC dataset . 124

8.10 semantic segmentation results on U-NET model in CityScape dataset. 125

8.11 Machine translation results on transformer model in WMT-2014 dataset. 126

8.12 Baseline table for TanhSoft-1, TanhSoft-2, and TanhSoft-3 based on all the experiments. The numbers represents the total number of models in which TanhSoft-1, TanhSoft-2, and TanhSoft-3 outperforms, equal or underperforms when compared to baseline activation functions . . . 127

8.13 Runtime comparison for the forward and backward passes for TanhSoft-1, TanhSoft-2, and TanhSoft-3 and baseline activation functions for a 32$\times$ 32 RGB image in ResNet-34 model. 128

9.1 Baseline table for EIS-1, EIS-2, and EIS-3. The integers represents the total number of models in which EIS-1, EIS-2, and EIS-3 outperforms, equal or underperforms when compared to baseline activations . . . 133

9.2 Results on MNIST, Fashion-MNIST and SVHN Datasets. 135

9.3 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on image classification problem on CIFAR10 dataset based on top-1 test accuracy. Top-1 accuracy(in %) for mean of 9 different runs have been reported. 136

9.4 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on image classification problem on CIFAR100 dataset based on top-1 test accuracy. Top-1 accuracy(in %) for mean of 9 different runs have been reported. 136

9.5 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Tiny ImageNet dataset on WRN 28-10 model. Results are reported for mean of 5 different runs. 138

9.6 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Object Detection problem on SSD 300 model on Pascal-VOC dataset. Results are reported for mean of 5 different runs. 138

9.7 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on semantic segmentation problem on U-NET model on Cityscapes dataset. Results are reported for mean of 5 different runs. 139

9.8 Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Machine translation problem on multi-head transformer model on WMT-2014 dataset. Results are reported for mean of 5 different runs. 139

9.9 Runtime comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 for the forward and backward passes for a 32× 32 RGB image on VGG-16 model. Results are reported for mean of 100 runs. 140

10.1 The relationship and difference between the proposed Activation functions and previously proposed widely used activation functions. . . . 142

10.2 Comparison between the proposed activations on the CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 5 different runs. mean±std is reported in the table . . 143

10.3 Comparison between the proposed activations on the Pascal VOC dataset for the object detection problem. We report the mAP for the mean of 3 different runs. mean±std is reported in the table. 144

10.4 Comparison between the proposed activations on the WMT2014 dataset for the machine translation problem. We report the BLEU score for the mean of 3 different runs. mean±std is reported in the table. 144

10.5 Runtime comparison between the proposed activation functions for the forward and backward passes for a 224× 224 RGB image on the ResNet-18 model. Results are reported for a mean of 100 runs. Experiments are conducted on an NVIDIA RTX 3090 GPU with 24 GB RAM. . . . 145

LIST OF FIGURES

1.1 Some widely used fixed activation functions 5

1.2 Some widely used trainable activation functions 6

3.1 Approximation of Leaky ReLU ($\alpha = 0.25$) using SAU. The left figure shows that SAU approximate Leaky ReLU smoothly, and in the right figure, we plot the same functions on a larger domain range. 15

3.2 Top-1 Train and Test accuracy Curves (Higher is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model. 21

3.3 Top-1 Train and Test Loss Curves (Lower is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model. 21

4.1 Approximation of ReLU using SMU ($\alpha = 0$) for different values of μ. As $\mu \to \infty$, SMU smoothly approximate ReLU 35

4.2 Approximation of Leaky ReLU ($\alpha = 0.25$) using SMU for different values of μ. As $\mu \to \infty$, SMU smoothly approximate Leaky ReLU . 35

4.3 First order derivatives of SMU for $\alpha = 0.25$ and different values of μ. 35

4.4 Approximation of ReLU using SMU-1 ($\alpha = 0$) for different values of μ. As $\mu \to 0$, SMU-1 smoothly approximate ReLU 35

4.5 Approximation of Leaky ReLU ($\alpha = 0.25$) using SMU-1 for different values of μ. As $\mu \to 0$, SMU-1 smoothly approximate Leaky ReLU 35

4.6 First order derivatives of SMU-1 for $\alpha = 0.25$ and different values of μ. 35

4.7 Approximation by a smooth function of $|x|$ 36

4.8 Top-1 train and test accuracy curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model. 46

4.9 Top-1 train and test loss curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model. 46

5.1 Swish and ErfAct activation for different values of α and β 59

5.2 Swish and Pserf activation for different values of γ and δ 59

5.3 First order derivative of ErfAct, Pserf, and Swish 59

5.4 Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations, ErfAct, and Pserf. 70

5.5 Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations, ErfAct, and Pserf. 70

6.1 Approximation of Leaky ReLU ($\alpha = 0.25$) and Leaky ReLU ($\alpha = 0.1$) using MAU. 77

6.2 Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations and the proposed activations. 85

6.3 Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations and the proposed activations. 85

7.1 Appximation of Leaky ReLU ($\alpha = 0.25$) by HP-1 function. 95

7.2 Appximation of Leaky ReLU ($\alpha = 0.25$) by HP-2 function. 95

7.3 Top-1 Train and Test accuracy (higher is better) on CIFAR10 dataset with LeNet model for different activations 104

7.4 Top-1 Train and Test loss (lower is better) on CIFAR10 dataset with LeNet model for for different activations 104

7.5 Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with MobileNet V2 network for different activations 108

7.6 Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with MobileNet V2 network for different activations 108

8.1 Plots of $\mathcal{F}_1(x;\alpha)$ for different values of α. 115

8.2 Plots of $\mathcal{F}_2(x;\beta,\gamma)$ for different values of β,γ. 115

8.3 Plots of $\mathcal{F}_3(x;\delta)$ for different values of δ. 115

8.4 Plots of $\mathcal{F}_1(x;0.87)$, $\mathcal{F}_2(x;0.75,0.75)$, $\mathcal{F}_3(x;0.85)$ and Swish. . . . 115

8.5 Plots of first derivative of $\mathcal{F}_1(x;\alpha)$ for different values of α. 116

8.6 Plots of first derivative of $\mathcal{F}_2(x;\beta,\gamma)$ for different values of β,γ. . . 116

8.7 Plots of first derivative of $\mathcal{F}_3(x;\delta)$ for different values of δ. 116

8.8 Plots of first order derivatives of $\mathcal{F}_1(x;0.87)$, $\mathcal{F}_2(x;0.75,0.75)$, $\mathcal{F}_3(x;0.85)$ and Swish. 116

8.9 Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3. 122

8.10 Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3. 122

8.11 Top-1 Train and Test accuracy (higher is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3. 123

8.12 Top-1 Train and Test loss (lower is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3. 123

9.1 Graph of $\mathcal{F}_1(x;\alpha,\beta)$ for different values of α,β. 131

9.2 Graph of $\mathcal{F}_2(x;\gamma)$ for different values of γ. 131

9.3 Graph of $\mathcal{F}_3(x;\delta,\theta)$ for different values of δ,θ. 131

9.4 Graph of first derivative of $\mathcal{F}_1(x;\alpha,\beta)$ for different values of α,β. . 131

9.5 Graph of first derivative of $\mathcal{F}_2(x;\gamma)$ for different values of γ. 131

9.6 Graph of first derivative of $\mathcal{F}_3(x;\delta,\theta)$ for different values of δ,θ. . . 131

9.7 Graph of Swish, $\mathcal{F}_1(x;\alpha,\beta)$, $\mathcal{F}_2(x;\gamma)$ and $\mathcal{F}_3(x;\delta,\theta)$ 132

9.8 Graph of first order derivatives of Swish, $\mathcal{F}_1(x;\alpha,\beta)$, $\mathcal{F}_2(x;\gamma)$, and $\mathcal{F}_3(x;\delta,\theta)$ 132

9.9 Graph for train and test accuracy on CIFAR100 dataset on WideResNet 28-10 model 137

9.10 Graph for train and test loss on CIFAR100 dataset on WideResNet 28-10 model 137

ABBREVIATIONS

AF	Activation Function
ANN	Artificial Neural Network
BN	Batch Normalization
CNN	Convolutional neural network
ELU	Exponential Linear Unit
erf	Gaussian error function
Leaky ReLU	Leaky Rectified Linear Unit
lr	Learning Rate
MAU	Maximum Activation Unit
OPAU	Orthogonal Padé Activation Unit
PAU	Padé Activation Unit
PReLU	Parametric Rectified Linear Unit
ReLU	Rectified Linear Unit
SAU	Smooth Activation Unit
SiLU	Sigmoid-weighted linear unit
SMU	Smooth Maximum Unit

CHAPTER 1

INTRODUCTION

Deep artificial neural networks (ANNs) are made up of several hidden layers, while each hidden layer consists of several neurons. At the level of each neuron, an affine linear map is composed with a nonlinear function known as activation function. During the training of an ANN, the linear map is optimized; however, an activation function is usually fixed in the beginning, along with the architecture of the ANN. There has been an increasing interest in developing a methodical understanding of activation functions, particularly with regards to the construction of novel activation functions and identifying mathematical properties leading to better learning (Nwankpa *et al.* (2018)).

Mathematically speaking, an activation function is a nonlinear mapping $f : \mathbb{R} \to \mathbb{R}$. A choice of such an f is considered good if it can generalise well on a variety of datasets, ensure faster convergence and improve neural network performance, which leads to more accurate results. During the early stages of deep learning research, shallow networks (fewer hidden layers) were used, along with tanh or sigmoid as activation functions. However as time progressed, and ANNs found more and more success in various fields of scientific research as well as in real life applications, and the need for deeper networks (more hidden layers) arose. The research also progressed and such networks came into fashion to achieve challenging tasks.

Deep neural networks has occupied the center-stage in modern machine learning research and application. And the activation functions, introducing non-linearity in the network can be considered to be the brain of the neural network. Consequently, the choice of activation function in a deep network can have a central role and significant impact on the performance, effectiveness, and training dynamics of deep neural networks.

Consequently, a significant amount of research has been dedicated to design better activation function in the recent years. The central focus of this thesis too is to construct novel activation functions that outperform the traditional used functions. We propose several novel activation functions, most of which consists of trainable parameters and

exhibit how they perform better than the popular ones in several metrics. All activation functions we construct are infinitely differentiable (smooth).

1.1 Related Works

Designing a new novel activation function is a difficult task. The machine learning community has so far relied on hand-designed activations like ReLU (Nair and Hinton (2010)), Leaky ReLU (Maas *et al.* (2013*a*)) or their variants. ReLU, in particular, remains widely popular due to faster training times and decent performance. However, evidence suggests that considerable gains can be made when more sophisticated activation functions are used to design networks. For example, activation functions such as ELU (Clevert *et al.* (2016)), Parametric ReLU (PReLU) (He *et al.* (2015*b*)), ReLU6 (Krizhevsky (2010)), PAU (Molina *et al.* (2020)), ACON (Ma *et al.* (2021)), Mish (Misra (2020)), GELU (Hendrycks and Gimpel (2020)), Swish (Ramachandran *et al.* (2017)) etc. have appeared as powerful contenders to the traditional ones. Though ReLU remains a go-to choice in both research and practice, it has certain well-documented shortcomings such as non-zero mean (Clevert *et al.* (2016)), non-differentiability and negative missing, which leads to the infamous vanishing gradients problem (also known as the dying ReLU problem). Worth noting that prior to the introduction of ReLU, Tanh and Sigmoid were popularly used, but performance gains and training time gains achieved by ReLU led to their decline.

Swish, GELU, Mish, and PAU are a few recently proposed activations, which gained popularity in the deep learning community. They share similar mathematical properties like smoothness, non-linearity, non-monotonic, small and bounded negative output. GELU is a popular activation widely used in Natural language processing tasks and recently used in BERT (Devlin *et al.* (2018)), GPT-2 (Radford *et al.* (2019)), and GPT-3 (Brown *et al.* (2020)) architectures. Swish was found by a group of researchers from Google by automated neural architecture search and shown promising results compared to ReLU. Mish is recently proposed by Misra, which shown some promising results on computer vision problems, especially on object detection task in YOLO v4 (Bochkovskiy *et al.* (2020)) model. PAU has been proposed recently, and it is constructed from the approximation of the Leaky ReLU function by rational polynomials

of a given order. Though PAU improves network performance in the image classification problem over ReLU, its variants, and Swish, it has a major drawback. PAU contains many trainable parameters, which significantly increases the network complexity and computational cost.

Motivated by these activation functions, we are interested in constructing some activations which share similar properties like the widely used activations. Also, they provides better performance in a wide range of deep learning problems (like Image classification, Object Detection, Semantic Segmentation, Machine Translation etc.) on different datasets and models when compared to widely used activations like ReLU, Leaky ReLU, Swish, GELU, PAU, and Mish.

1.1.1 Types of Activation Functions:

Activation functions can broadly be classified into two types: *fixed activation functions* and *trainable activation functions*.

Fixed activation functions: Fixed activations are handcrafted activations and fixed before training. They may contain constant hyperparameters but does not contain any trainable parameters. The following are some well established fixed activation functions.

1. **Sigmoid:**Sigmoid is a nonlinear activation function and widely used in feed-forward network in internal networks before ReLU had been proposed. Now, it is used in output-layer for binary classification problems. Sigmoid is defined as

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

   It has a major drawback that it ranges in 0 to 1 which sometime leads to vanishing gradient problem.

2. **tanh:** The hyperbolic tangent, or tanh function was also used before ReLU proposed as activation function. tanh is defined as

$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

   tanh ranges between -1 to 1.

3. **ReLU:** ReLU (Nair and Hinton (2010)) is defined as

$$f(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

ReLU is popular due to its simplicity though has some drawbacks like "Dying ReLU" (Maas *et al.* (2013*b*)) problem (It happens when a large number of neurons produce zero output irrespective of any inputs) or non-differentiable at zero.

4. **Leaky ReLU:** Leaky ReLU (Maas *et al.* (2013*a*)) is a variant of ReLU where a non-negative component has been introduced to overcome the "Dying ReLU" problem. Leaky ReLU is defined as

$$f(x;a) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \end{cases}$$

where a is constant hyperparameter. Leaky ReLU increases the network performance marginally compared to ReLU..

5. **ReLU6:** ReLU6 (Krizhevsky (2010)) is a special case of ReLU where the maximum value of the function is 6 and the function is defined as

$$f(x) = \begin{cases} 0 & \text{if } x < 0 \\ x & \text{if } 0 \leq x < 6 \\ 6 & \text{if } x \geq 6 \end{cases}$$

6. **ELU:** ELU (Clevert *et al.* (2016)) is a piecewise activation function defined as

$$f(x;a) = \begin{cases} x & \text{if } x \geq 0 \\ a(e^x - 1) & \text{if } x < 0 \end{cases}$$

a is a hyperparameter controls the values in the negative axis. ELU try to keep the mean towards zero which leads to faster learning (see (Clevert *et al.* (2016)) for more details).

7. **Softplus:** Softplus (Zheng *et al.* (2015)) is a smooth activation function and can be viewed as a approximation by a smooth function of ReLU. Softplus is defined as

$$f(x) = ln(1 + e^x)$$

8. **Sigmoid-weighted linear unit:** Sigmoid-weighted linear unit (SiLU) (Elfwing *et al.* (2017)) is a smooth activation function. It is the product of linear function with sigmoid function. Formally, it is defined as

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

SiLU increases the network performance significantly compared to ReLU.

9. **GELU:** Gaussian Error Linear Unit (GELU) (Hendrycks and Gimpel (2020)) is a non-linear activation function which can be seen as a approximation by a smooth function of ReLU. GELU is defined as

$$f(x) = x.\phi(x) = \frac{x}{2}(1 + \text{erf}(\frac{x}{\sqrt{2}}))$$

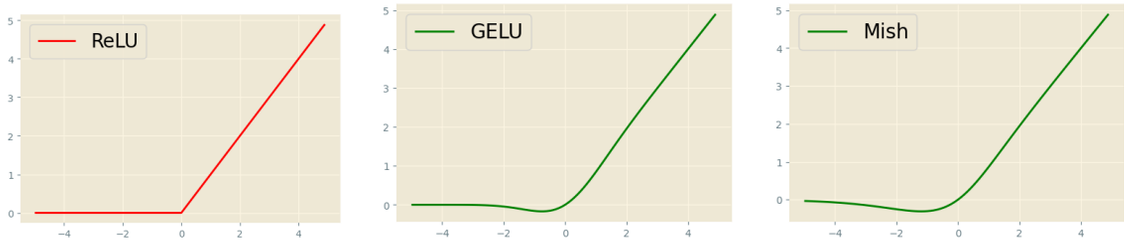

Figure 1.1: Some widely used fixed activation functions

where $\phi(x)$ is the cumulative Gaussian distribution function and erf is the Gaussian error function defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

GELU can be approximated by

$$f(x) = 0.5x \left(1 + \tanh\left[\sqrt{2/\pi}\left(x + 0.044715x^3\right)\right]\right)$$

GELU improves the performance of ReLU and widely used in natural language processing architectures.

10. **Mish:** Mish (Misra (2020)) is a popular activation function proposed recently and is defined as

$$f(x) = x \tanh(ln(1 + e^x))$$

Mish is a smooth non-monotonic, non-linear activation function which usually provide better performance compared to ReLU and Swish.

Trainable Activation functions: Trainable activation function contains trainable parameter(s) which is initialized at a point before training and updated during backpropagation. The following are some well established trainable activation functions.

1. **Parametric ReLU:** Parametric ReLU (PReLU) (He *et al.* (2015*b*)) is the trainable form of Leaky ReLU. PReLU is defined as

$$f(x; a) = \begin{cases} x & \text{if } x \geq 0 \\ ax & \text{if } x < 0 \end{cases}$$

where a is a trainable parameter. Observe that for $a = 0$, it is ReLU and if a is constant, then it is Leaky ReLU. PReLU is continuous, unbounded but non-differentiable at zero.

2. **Swish:** Swish (Ramachandran *et al.* (2017)) is a popular activation function proposed by Google brain team by neural architecture search. Swish is defined as

$$f(x) = \frac{x}{1 + e^{-\beta x}}$$

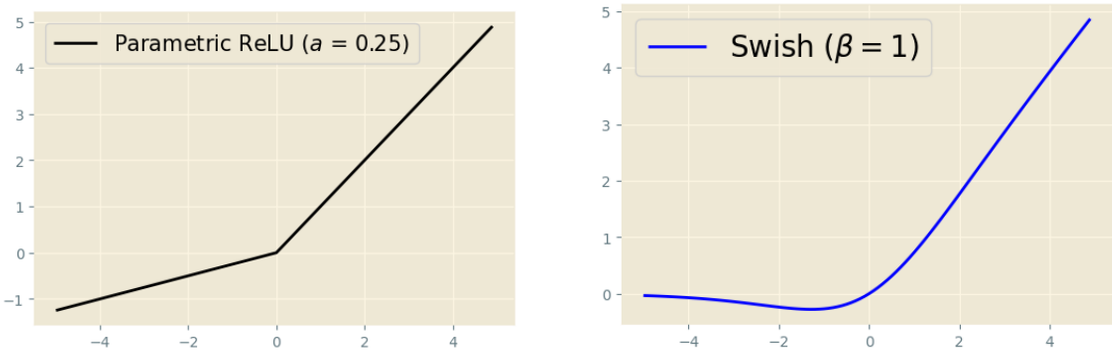

Figure 1.2: Some widely used trainable activation functions

Observe that as $\beta \to \infty$, Swish $\to$ ReLU. So Swish can be seen as a approximation by a smooth function of ReLU. Swish outperforms ReLU on variety of large datasets in large models in different deep learning problems. Swish has a good potentials to replace ReLU.

3. **Padé Activation Unit:** Padé Activation Unit (PAU) (Molina *et al.* (2020)) has been proposed recently and it is defined as the approximation of known activation function by rational polynomial approximation known as Padé approximation. PAU is defined as the approximation of Leaky ReLU using the following form

$$F_2(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} a_i x^i}{1 + |\sum_{j=1}^{l} b_j x^j|} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k}{1 + |b_1 x + b_2 x^2 + \cdots + b_l x^l|}$$

1.2 Research Contributions

As discussed in the previous section, two types of activations gain attention from the research community: (i) Fixed activation function and (ii) trainable activation function. To on this effect, this thesis primarily focuses on designing new novel activations which performs better than the widely used activations such as ReLU, Leaky ReLU, Swish etc. in standard deep learning problems. All the activation that has been proposed in the subsequent chapters are either designed from some underlying approximation theory or handcrafted. The proposed functions can be used as trainable activation functions or fixed activation functions. The proposed activations have been widely tested on standard publicly available benchmark datasets, and comparisons have been done on the state-of-the-art models with the widely used activations. The main contributors to this thesis are as follows:

- **Smooth Maximum Unit:** ReLU or its variants are non-smooth at the origin. The functional form of these functions contains absolute function which is non-

differentiable at the origin. We have replaced the absolute function by two functions which have been approximated by a smooth function. We show that the resulting function is smooth. It can approximate general maxout (Goodfellow *et al.* (2013)) family, ReLU or its variants. We also show that GELU is a special case of our proposal. Experimentally, we show that the proposed functions outperforms widely used functions in bench-marking datasets on standard deep learning problems.

- **Smooth Activation Unit:** As discussed earlier, ReLU and its variants are non-smooth at the origin. Approximation by a smooth function using mollifier and approximate identities, an activation has been proposed. This function can smoothly approximate ReLU or its variants. Experimentally it has been shown that the function outperforms widely used activations in standard deep learning problems on bench-marking datasets.

- **Maximum Activation Unit:** This chapter presents that the maximum function can be written in a special form and then it has been shown that we can smoothly approximate that special form by some approximation formula. Three smooth activations have been proposed in this chapter. Experimentally, it has been found that the propose functions outperforms widely used activations in standard deep learning problems on bench-marking datasets.

- **Orthogonal-Padé Activation Unit:** Padé approximation is a well known approximation by rational function approximation using polynomials. The polynomials have been replaced in both numerator and denominator with some widely used popular orthogonal polynomials and a special form is proposed. The form can approximate any continuous function. Leaky ReLU is approximated by orthogonal Padé form and the approximated form is a approximation by a smooth function, it has been named as Orthogonal-Padé Activation Unit (OPAU). Six orthogonal polynomial is considered to test their performance. Hermite polynomial form comes out to be the best choice. Experimentally, it has been found that the propose functions outperforms widely used activations in standard deep learning

problems on bench-marking datasets.

- **ErfAct and Pserf:** This chapter presents two activations which is called ErfAct and Pserf. These functions are handcrafted functions like other widely used activation's. Both the functions are constructed using the Gaussian error function. It is shown that both the functions are approximation by a smooth function of ReLU. Experimentally, it has been found that the propose functions outperforms widely used activations in standard deep learning problems on bench-marking datasets.

- **TanhSoft:** This chapter presents three handcrafted activations which are named as TanhSoft-1, TanhSoft-2, and TanhSoft-3. These functions are constructed using the tanh function. Experimentally, it has been found that the propose functions outperforms widely used activations in standard deep learning problems on bench-marking datasets.

- **EIS:** This chapter presents three handcrafted activations which are named as EIS-1, EIS-2, and EIS-3. Experimentally, it has been found that the propose functions outperforms widely used activations in standard deep learning problems on bench-marking datasets.

Each subsequent chapters in this thesis provides details about construction and analysis of the above mentioned activation functions, along with data from several experiments.

CHAPTER 2

Proposed Problems and Methodology

2.1 Proposed problems

This thesis addresses three points: (1) Proposing novel activation functions that outperform the existing activation functions, (2) Making methodical perturbations of modifications to known activation function to understand how it affects performance, and (3) Exploring through automated parameter search, extensive classes of functions and their suitability as activation functions.

There is increasing interest in designing new neural network architecture to boost network performance in the deep learning domain. An important part of this problem is the search for novel activation functions, which in recent times has evolved into a fundamental research problem in deep learning, and the domain requires further exploration. An activation function is the core component of a neural network that introduces non-linearity. In the early '90s, tanh and sigmoid were widely used activation functions. Once ReLU was proposed, it gained attention from the deep learning community and became the default activation for the neural network due to its simplicity. However, ReLU has some serious drawbacks (non-smooth, negative missing etc.) and to overcome them, researchers have come up with more sophisticated smooth activation functions. Two such activation functions proposed only a few years ago, namely, Swish (Ramachandran *et al.* (2017)) and GELU (Hendrycks and Gimpel (2020)), have seen an increase in popularity in the community. These, in a certain sense, can be considered to be an approximation by a smooth function of the ReLU function, and their usage boosts network performance compared to ReLU in various deep learning tasks. Motivated by these new developments in network design via the construction of more evolved activation functions, this thesis tries to develop novel activation functions that perform better than the other widely used ones such as ReLU, Swish, ELU, SoftPlus, GELU, PAU, etc.

The thesis proposes activation functions in two different ways: handcrafted and via the usage of approximation methods. This thesis not only concentrates on constructing

activation functions using the brute force method but also on constructing activation functions by approximating known activation functions and using various well-known approximations to the general maximum family. The performance of the proposed activation functions are evaluated on standard and widely used datasets on four fundamental and different deep learning problems: image classification, object detection, semantic segmentation, and machine translation. The performance of the proposed activation functions is compared with widely used functions like ReLU, Leaky ReLU, Parametric ReLU, Mish, GELU, Swish, Softplus etc.

2.2 Methodology:

The efficacy of the proposed activation functions are established via empirical evaluation on four different fundamental deep learning problems: image classification, object detection, semantic segmentation, and machine translation.

For experiments, a comprehensive list of benchmarking datasets was considered. MNIST, Fashion MNIST, SVHN, CIFAR10, CIFAR100, and Tiny Imagenet were used for image classification problems. Pascal VOC is considered for object detection problems. CityScapes is considered for the semantic segmentation problem. WMT2014 has been considered for the machine translation problem. All the datasets mentioned above have been historically used for the four important problems mentioned earlier.

We follow the same standard methodology & experimental procedure (problem and datasets) that previously established popular works like Swish, PAU etc., considered to establish their method. We also report and compare the running time for each proposed activation function. To compare the performance and efficiency of our proposed activation function, we report results with other widely used activations like ReLU, Leaky ReLU, ELU, SoftPlus, GELU, Swish etc., for each dataset.

CHAPTER 3

Smooth Activation Unit[1]

3.1 Introduction

As described in the previous two introductory chapters, two primary methods for constructing novel activation are being explored in this thesis: handcrafting activation functions and perturbation of known activation functions. This chapter, in particular, deals with the later of the two methods. The ReLU or Leaky ReLU is methodically perturbed to smooth activations, and yet these functions lie close to the original activations from which they were obtained.

Deep networks form a crucial component of modern deep learning. Non-linearity is introduced in such networks by using activation functions, and the choice substantially impacts network performance and training dynamics. Designing a new novel activation function is a difficult task. ReLU remains the favourite choice among the deep learning community due to its simplicity and better performance when compared to Tanh or Sigmoid. However, it has a drawback known as dying ReLU, in which the network starts to lose the gradient direction due to the negative inputs and produces zero outcomes. In 2017, Swish (Ramachandran *et al.* (2017)) was proposed by the Google brain team. Swish was found by automatic search technique, and it has shown some promising performance across different deep learning tasks.

3.2 Related works and Motivation

Handcrafted activations like Rectified Linear Unit (ReLU) (Nair and Hinton (2010)), Leaky ReLU (Maas *et al.* (2013*a*)) or its variants are very common choices for activation functions and exhibits promising performance on different deep learning tasks. There are many activations that have been proposed so far. Some of them are ELU

[1]This chapter is a slightly modified version of the paper in Arxiv Biswas *et al.* (2021*d*).

(Clevert *et al.* (2016)), Parametric ReLU (PReLU) (He *et al.* (2015*b*)), Swish (Ramachandran *et al.* (2017)), Padé Activation Unit (PAU) (Molina *et al.* (2020), ACON Ma *et al.* (2021), Mish (Misra (2020)), GELU (Hendrycks and Gimpel (2020)), ReLU6 (Krizhevsky (2010)), Softplus (Zheng *et al.* (2015)) etc. Activation functions are usually handcrafted and fixed before training. PReLU (He *et al.* (2015*b*)) tries to overcome this problem by introducing a learnable negative component to ReLU (Nair and Hinton (2010)). Maxout (Goodfellow *et al.* (2013)) and Mixout (Hui-zhen Zhao (2017)) are constructed with piecewise linear components, and theoretically, they are universal function approximators, though they increase the number of parameters in the network. Recently, meta-ACON (Ma *et al.* (2021)), a smooth activation, has been proposed, which is the generalization of the ReLU and Maxout activations and can smoothly approximate Swish. Meta-ACON has shown some good improvement on both small models and highly optimized large models. PAU (Molina *et al.* (2020)) is a promising candidate for trainable activations, which have been introduced recently based on rational function approximation. However, PAU contains a large number of parameters which increase the computational complexity of the network.

3.3 Research Contribution

In this chapter, we introduce a smooth approximation of known non-smooth activation functions like ReLU or Leaky ReLU based on the approximation of identity. To validate the performance of the proposed activation function, a wide range of experiments have been conducted on four important and different deep learning problems like image classification, object detection, semantic segmentation, and machine translation. The results are reported in the experiment section.

3.4 Mathematical formalism

3.4.1 Convolution

Convolution is a binary operation, which takes two functions f and g as input, and outputs a new function denoted by $f * g$. Mathematically, we define this operation as

follows

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y)g(x-y)\,dy. \tag{3.1}$$

The convolution operation has several properties. Below, we will list two of them which will be used larter in this article.

P1. $(f * g)(x) = (g * f)(x)$,

P2. If f is n-times differentiable with compact support over $\mathbb{R}$ and g is locally integrable over $\mathbb{R}$ then $f * g$ is at least n-times differentiable over $\mathbb{R}$.

Property P1 is an easy consequence of definition (3.1). Property P2 can be easily obtained by moving the derivative operator inside the integral. Note that this exchange of derivative and integral requires f to be of compact support. An immediate consequence of property P2 is that if one of the functions f or g is smooth with compact support, then $f * g$ is also smooth. This observation will be used later in the article to obtain smooth approximations of non-differentiable activation functions.

3.4.2 Mollifier and Approximate identities

A smooth function ϕ over $\mathbb{R}$ is called a mollifier if it satisfies the following three properties:

1. It is compactly supported.
2. $\int_{\mathbb{R}} \phi(x)\,dx = 1$.
3. $\lim\limits_{\epsilon \to 0} \phi_\epsilon(x) := \lim\limits_{\epsilon \to 0} \frac{1}{\epsilon}\phi(x/\epsilon) = \delta(x)$, where $\delta(x)$ is the Dirac delta function.

We say that a mollifier ϕ is an approximate identity if for any locally integrable function f over $\mathbb{R}$, we have

$$\lim_{\epsilon \to 0}(f * \phi_\epsilon)(x) = f(x) \text{ pointwise for all } x.$$

3.4.3 Smooth approximations of non-differentiable functions

Let ϕ be an approximate identity. Choosing $\epsilon = 1/n$ for $n \in \mathbb{N}$, one can define

$$\phi_n(x) := n\phi(nx). \tag{3.2}$$

Using the property of approximate identity, for any locally integrable function f over $\mathbb{R}$, we have

$$\lim_{n\to\infty} (f * \phi_n)(x) = f(x) \text{ pointwise for all } x.$$

That is, for large enough n, $f * \phi_n$ is a good approximation of f. Moreover, since ϕ is smooth, ϕ_n is smooth for each $n \in \mathbb{N}$ and therefore, using property P2, $f * \phi_n$ is a smooth approximation of f for large enough n.

Let $\sigma : \mathbb{R} \to \mathbb{R}$ be any activation function. Then, by definition, σ is a continuous and hence, a locally integrable function. For a given approximate identity ϕ and $n \in \mathbb{N}$, we define a smooth approximation of σ as $\sigma * \phi_n$, where ϕ_n is defined in (3.2).

3.5 Smooth Activation Unit (SAU)

Consider the Gaussian function

$$\phi(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$$

which is a well known approximate identity. Consider the Leaky Rectified Linear Unit (Leaky ReLU) activation function

$$\text{LeakyReLU}[\alpha](x) = \begin{cases} x & x \geq 0 \\ \alpha x & x < 0 \end{cases}$$

Note that $\text{LeakyReLU}[\alpha]$ activation function is hyperparametrized by α and it is non-differentiable at the origin for all values of α except $\alpha = 1$. For $\alpha = 0$, $\text{LeakyReLU}[\alpha]$ reduces to well known activation function ReLU (Nair and Hinton (2010)) while for constant and trainable α, $\text{LeakyReLU}[\alpha]$ reduces to Leaky ReLU (Maas *et al.* (2013*a*))

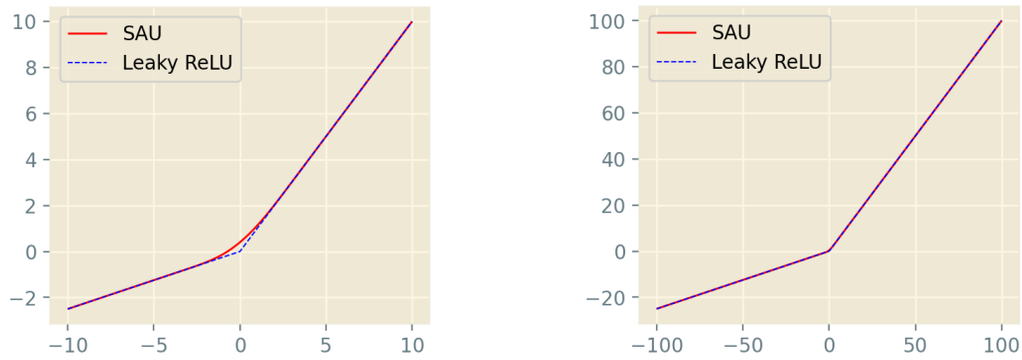

Figure 3.1: Approximation of Leaky ReLU ($\alpha = 0.25$) using SAU. The left figure shows that SAU approximate Leaky ReLU smoothly, and in the right figure, we plot the same functions on a larger domain range.

and Parametric ReLU (He *et al.* (2015*b*)) respectively. For a given $n \in \mathbb{N}$, and $\alpha \neq 1$, a smooth approximation of $\mathrm{LeakyReLU}[\alpha]$ is given by

$$G(x, \alpha, n) = (\mathrm{LeakyReLU}[\alpha] * \phi_n)(x) = \frac{1}{2n}\sqrt{\frac{2}{\pi}} e^{\frac{-n^2x^2}{2}} + \frac{(1+\alpha)}{2}x + \frac{(1-\alpha)}{2}x \ \mathrm{erf}\left(\frac{nx}{\sqrt{2}}\right) \tag{3.3}$$

where erf is the Gaussian error function

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

For the rest of the chapter, we will only consider the approximate identity of Leaky ReLU ($\alpha = 0.25$) given in equation 3.3 as the activation function. We call this function Smooth Activation Unit (SAU). Approximation of Leaky ReLU ($\alpha = 0.25$) by SAU is given in figure 3.1. It is clear from the figure 3.1 that SAU can smoothly approximate Leaky ReLU (as well as ReLU or its variants) quite well. We note that in GELU (Hendrycks and Gimpel (2020)) paper, the authors use the product of x with the cumulative distribution function of a suitable probability distribution (see (Hendrycks and Gimpel (2020)) for further details).

3.5.1 Learning activation parameters via back-propagation

Back-propagation algorithm (LeCun *et al.* (1989)) and gradient descent is used in neural networks to update Weights and biases. Parameters in trainable activation functions are

updated using the same technique. The forward pass is implemented in both Pytorch (Paszke *et al.* (2019)) & Tensorflow-Keras (Chollet *et al.* (2015)) API, and automatic differentiation will update the parameters. Alternatively, CUDA (Nickolls *et al.* (2008)) based implementation (see (Maas *et al.* (2013*a*))) can be used and the gradients of equation 3.3 for the input x and the parameter α & n can be computed as follows:

$$\frac{\partial G}{\partial x} = \frac{-nx}{2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} + \frac{(1+\alpha)}{2} + \frac{(1-\alpha)}{2}\ \text{erf}\left(\frac{nx}{\sqrt{2}}\right) + \frac{n(1-\alpha)}{\sqrt{2\pi}}\ x\ e^{-\frac{n^2x^2}{2}} \tag{3.4}$$

$$\frac{\partial G}{\partial \alpha} = \frac{x}{2}\left(1 - \text{erf}\left(\frac{nx}{\sqrt{2}}\right)\right). \tag{3.5}$$

$$\frac{\partial G}{\partial n} = -\frac{1}{2n^2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} - \frac{x^2}{2}\sqrt{\frac{2}{\pi}}e^{\frac{-n^2x^2}{2}} + \frac{x^2(1-\alpha)}{\sqrt{2\pi}}\ e^{-\frac{n^2x^2}{2}}. \tag{3.6}$$

where

$$\frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}$$

α and n can be either hyperparameters or trainable parameters.

Now, note that the class of neural networks with SAU activation function is dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K.

The proof follows from the following proposition (see (Molina *et al.* (2020))).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 (Kidger and Lyons (2020))) :- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

3.6 Experiments

To explore and compare the performance of SAU, we consider eight popular standard activation functions on different standard datasets and popular network architectures on standard deep learning problems like image classification, object detection, semantic segmentation, and machine translation. We consider the following activations to compare with SAU: ReLU, Leaky ReLU, Parametric ReLU (PReLU), ELU, ReLU6, Softplus, PAU, Swish, and GELU. It is evident from the experimental results in the next sections that SAU outperform in most cases compared to the standard activations. We consider α as a hyperparameter and n as a trainable parameter for the rest of our experiments. We fix α at 0.25. The value of n is considered 20000 and updated via backpropagation according to equation 3.6. All the experiments are conducted on an NVIDIA V100 GPU with 32GB RAM.

3.6.1 Image Classification

MNIST, Fashion MNIST and The Street View House Numbers (SVHN) Database:

In this section, we present results on MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), and SVHN (Netzer *et al.* (2011)) datasets. The MNIST and Fashion MNIST databases have a total of 60k training and 10k testing 28×28 grey-scale images with ten different classes. SVHN consists of 32×32 RGB images with a total of 73257 training images and 26032 testing images with ten different classes. We have applied standard data augmentation methods like rotation, zoom, height shift, shearing on the three datasets. We report results with LeNet (Lecun *et al.* (1998)), AlexNet (Krizhevsky *et al.* (2012)), and VGG-16 (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))) architecture in Table 3.1, Table 3.2, and Table 3.3 respectively. We report a more detailed experiment on MNIST, Fashion MNIST, and SVHN datasets on a custom-designed model in Table 3.4. We design the custom network with CNN layers with 3×3 kernels and max-pooling layers with 2×2 kernels. We consider Channel depths of size 128 (twice), 64 (thrice), 32 (twice), with a dense layer of size 128, Max-pooling layer(thrice), and dropout. We have applied

batch-normalization before the activation function layer. For all the experiments to train a model on these three datasets, we use a batch size of 128, stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 100 epochs. We begin with 0.01 learning rate and decay the learning rate with cosine annealing (Loshchilov and Hutter (2017)) learning rate scheduler.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.21 ± 0.10	91.51 ± 0.20	92.17 ± 0.19
Leaky ReLU	99.17 ± 0.10	91.61 ± 0.21	92.31 ± 0.18
PReLU	99.27 ± 0.09	91.62 ± 0.18	92.05 ± 0.21
ReLU6	99.29 ± 0.08	91.57 ± 0.17	92.25 ± 0.17
ELU	99.28 ± 0.10	91.48 ± 0.19	92.20 ± 0.18
Softplus	99.06 ± 0.16	91.21 ± 0.23	91.89 ± 0.25
PAU	99.34 ± 0.07	**91.69** ± 0.12	92.31 ± 0.22
Swish	99.31 ± 0.07	91.64 ± 0.14	92.39 ± 0.20
GELU	99.29 ± 0.06	91.61 ± 0.14	92.42 ± 0.20
SAU	**99.40** ± 0.05	91.47 ± 0.16	**92.61** ± 0.12

Table 3.1: A Detailed Comparison between SAU Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with LeNet Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.51 ± 0.06	92.77 ± 0.18	95.11 ± 0.14
Leaky ReLU	99.50 ± 0.06	92.79 ± 0.20	95.21 ± 0.17
PReLU	99.48 ± 0.08	92.76 ± 0.18	95.19 ± 0.17
ReLU6	99.55 ± 0.06	93.01 ± 0.16	95.22 ± 0.15
ELU	99.56 ± 0.05	92.89 ± 0.17	95.30 ± 0.18
Softplus	99.22 ± 0.10	92.32 ± 0.25	94.82 ± 0.21
PAU	99.53 ± 0.08	93.01 ± 0.17	95.22 ± 0.13
Swish	99.58 ± 0.06	92.96 ± 0.16	95.32 ± 0.14
GELU	99.55 ± 0.06	93.05 ± 0.14	95.28 ± 0.14
SAU	**99.64** ± 0.04	**93.17** ± 0.14	**95.45** ± 0.11

Table 3.2: A Detailed Comparison between SAU Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with AlexNet Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.55 ± 0.07	93.75 ± 0.14	96.04 ± 0.12
Leaky ReLU	99.59 ± 0.05	93.89 ± 0.14	96.12 ± 0.15
PReLU	99.58 ± 0.07	93.85 ± 0.16	96.12 ± 0.17
ReLU6	99.59 ± 0.05	93.88 ± 0.11	96.18 ± 0.16
ELU	99.51 ± 0.05	93.82 ± 0.16	96.13 ± 0.14
Softplus	99.34 ± 0.12	93.69 ± 0.19	95.88 ± 0.21
PAU	99.58 ± 0.05	94.27 ± 0.12	96.20 ± 0.15
Swish	99.54 ± 0.06	94.10 ± 0.12	96.26 ± 0.13
GELU	99.60 ± 0.04	94.17 ± 0.12	96.23 ± 0.13
SAU	$\mathbf{99.67} \pm 0.04$	$\mathbf{94.40} \pm 0.12$	$\mathbf{96.41} \pm 0.12$

Table 3.3: A Detailed Comparison between SAU Activation and Other Baseline Activations On MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with VGG16 Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.48 ± 0.07	93.06 ± 0.14	94.52 ± 0.13
Leaky ReLU	99.43 ± 0.08	93.22 ± 0.15	94.67 ± 0.16
PReLU	99.41 ± 0.09	93.14 ± 0.12	94.61 ± 0.15
ReLU6	99.45 ± 0.07	93.31 ± 0.14	94.70 ± 0.10
ELU	99.50 ± 0.05	93.28 ± 0.16	94.62 ± 0.13
Softplus	99.32 ± 0.10	92.95 ± 0.20	94.41 ± 0.18
PAU	$\mathbf{99.58} \pm 0.09$	93.26 ± 0.16	94.89 ± 0.11
Swish	99.57 ± 0.08	93.29 ± 0.14	94.72 ± 0.12
GELU	99.50 ± 0.05	93.35 ± 0.12	94.79 ± 0.09
SAU	99.57 ± 0.04	$\mathbf{93.50} \pm 0.09$	$\mathbf{95.10} \pm 0.09$

Table 3.4: A Detailed Comparison Between SAU Activation and Other Baseline Activations in MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem on Custom-designated Architecture. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

CIFAR:

The CIFAR (Krizhevsky (2009)) is one of the most popular databases for image classification consists of a total of 60k 32×32 RGB images and is divided into 50k training and 10k test images. CIFAR has two different datasets- CIFAR10 and CIFAR100 with a total of 10 and 100 classes, respectively. We report the top-1 accuracy on Table 3.5 and Table 3.10 on CIFAR100 and CIFAR10 datasets respectively. We consider MobileNet V1 (Howard *et al.* (2017)), MobileNet V2 (Sandler *et al.* (2019)), Shufflenet V2 (Ma *et al.* (2018)), PreActResNet (He *et al.* (2016)), ResNet (He *et al.* (2015*a*)), Inception V3 (Szegedy *et al.* (2015*a*)), squeeze and excitation networks (SeNet) (Hu *et al.* (2017)), ResNext (Xie *et al.* (2017)), LeNet (Lecun *et al.* (1998)), AlexNet (Krizhevsky *et al.* (2012)), DenseNet (Huang *et al.* (2016*a*)), Xception (Chollet (2017)), Squeezenet (Iandola *et al.* (2016)), WideResNet (Zagoruyko and Komodakis (2016)), VGG (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))), and EfficientNet B0 (Tan and Le (2020)). For all the experiments to train a model on these two datasets, we use a batch size of 128, stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. We begin with 0.01 learning rate and decay the learning rate by a factor of 10 after every 60 epochs. Standard data augmentation methods like width shift, height shift, horizontal flip, and rotation is applied on both datasets. It is noticeable from these two tables that replacing ReLU by SAU, there is an increment in top-1 accuracy from 1% to more than 5% in most of the models. A more detailed result on these two datasets with other baseline activations are reported Tables 3.7, 3.8, 3.9, 3.10, 3.11, and 3.12. Training and test accuracy & loss curves for baseline activation functions and SAU are given in Figures 3.2 and 3.3 respectively on CIFAR100 dataset on ShuffleNet V2 (2.0x) network. From these learning curves, it is evident that after training few epochs, SAU has stable & smooth learning, faster convergence speed, and higher accuracy and lower loss on the test dataset compared to other baseline activation functions.

Also, We compare the performance of SAU with other baseline activations with state of the art data augmentation method like Mixup (Zhang *et al.* (2017*a*)) on CIFAR 100 dataset with ShuffleNet V2 (2.0x), ResNet 18 & ResNet 50 models, and we got very good improvement over the baseline activations. Results are reported on Table 3.13

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
Shufflenet V2 0.5x	61.76 ± 0.27	**64.39** ± 0.23
Shufflenet V2 1.0x	64.12 ± 0.28	**68.41** ± 0.24
Shufflenet V2 1.5x	66.52 ± 0.28	**71.97** ± 0.24
Shufflenet V2 2.0x	66.94 ± 0.24	**72.57** ± 0.21
PreActResNet 18	72.58 ± 0.24	74.01 ± 0.22
PreActResNet 34	72.92 ± 0.24	**75.37** ± 0.24
PreActResNet 50	73.27 ± 0.25	**76.22** ± 0.22
ResNet 18	73.02 ± 0.25	74.27 ± 0.22
ResNet 34	73.12 ± 0.26	**74.64** ± 0.23
ResNet 50	73.89 ± 0.23	**76.39** ± 0.20
MobileNet V1	70.95 ± 0.26	**72.09** ± 0.23
MobileNet V2	73.85 ± 0.24	**75.69** ± 0.19
Inception V3	74.03 ± 0.27	**76.01** ± 0.22
WideResNet 28-10	75.89 ± 0.23	**77.39** ± 0.20
DenseNet 121	75.72 ± 0.27	**77.11** ± 0.23
EffitientNet B0	76.22 ± 0.24	**78.07** ± 0.26
VGG16	71.10 ± 0.30	71.18 ± 0.28

Table 3.5: A Detailed Comparison between SAU Activation and Other Baseline Activations on The CIFAR100 Dataset for Image Classification Problem with Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

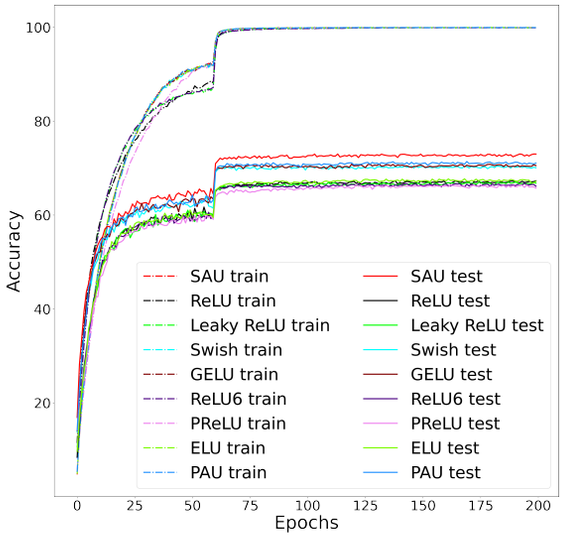

Figure 3.2: Top-1 Train and Test accuracy Curves (Higher is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model.

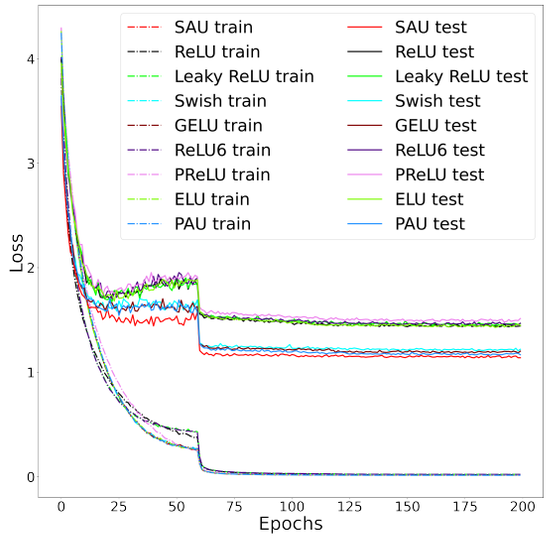

Figure 3.3: Top-1 Train and Test Loss Curves (Lower is Better) for SAU and Baseline Activation Functions on CIFAR100 Dataset with ShuffleNet V2 (2.0x) Model.

Model	ReLU	SAU
	Top-1 accuracy (mean± std)	Top-1 accuracy (mean ± std)
ShuffleNet V2 0.5x	88.01 ± 0.23	**90.50** ± 0.17
ShuffleNet V2 1.0x	90.74 ± 0.25	**92.78** ± 0.20
ShuffleNet V2 1.5x	91.07 ± 0.23	**93.20** ± 0.18
ShuffleNet V2 2.0x	91.32 ± 0.22	**93.52** ± 0.16
PreActResNet 18	93.36 ± 0.18	94.62 ± 0.15
PreActResNet 34	94.01 ± 0.16	**95.10** ± 0.14
PreActResNet 50	94.01 ± 0.15	**94.94** ± 0.14
ResNet 18	93.32 ± 0.20	93.47 ± 0.17
ResNet 34	93.77 ± 0.20	**94.22** ± 0.16
ResNet 50	93.89 ± 0.19	**94.62** ± 0.16
MobileNet V1	92.27 ± 0.24	**93.54** ± 0.14
MobileNet V2	93.89 ± 0.19	**95.37** ± 0.09
Inception V3	93.89 ± 0.18	**94.51** ± 0.10
WideResNet 28-10	94.74 ± 0.18	**95.52** ± 0.12
DenseNet 121	94.41 ± 0.16	**95.31** ± 0.10
EffitientNet B0	94.64 ± 0.16	**95.52** ± 0.14
VGG16	93.14 ± 0.23	93.31 ± 0.21

Table 3.6: A Detailed Comparison between SAU Activation and Other Baseline Activations on The CIFAR10 Dataset for Image Classification Problem with Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MobileNet V1	MobileNet V2	ResNet 18	ResNet 34	ResNet 50
Leaky ReLU	88.11 ±0.24	90.85 ±0.27	91.02 ±0.23	91.44 ±0.24	92.40 ±0.26	93.78 ±0.21	93.22 ±0.21	93.89 ±0.21	93.78 ±0.20
PReLU	88.17 ±0.24	90.88 ±0.27	91.19 ±0.24	91.39 ±0.22	92.44 ±0.23	94.08 ±0.22	93.20 ±0.23	93.79 ±0.22	93.91 ±0.20
ReLU6	88.23 ±0.22	90.89 ±0.24	91.09 ±0.21	91.57 ±0.22	92.49 ±0.24	93.89 ±0.20	93.37 ±0.21	93.98 ±0.20	93.85 ±0.22
ELU	88.22 ±0.23	90.84 ±0.26	91.17 ±0.24	91.52 ±0.23	92.52 ±0.25	93.89 ±0.22	93.32 ±0.23	94.01 ±0.21	93.80 ±0.21
Softplus	87.86 ±0.26	90.52 ±0.28	91.01 ±0.27	91.42 ±0.24	92.32 ±0.29	93.95 ±0.24	93.31 ±0.24	93.59 ±0.24	93.70 ±0.23
PAU	88.86 ±0.24	91.55 ±0.26	92.45 ±0.19	92.71 ±0.22	92.67 ±0.17	94.68 ±0.19	93.79 ±0.17	94.01 ±0.20	94.12 ±0.16
Swish	88.67 ±0.23	91.72 ±0.25	92.21 ±0.20	92.45 ±0.21	92.52 ±0.19	94.78 ±0.21	93.71 ±0.18	94.10 ±0.19	94.24 ±0.18
GELU	88.51 ±0.22	91.81 ±0.23	92.59 ±0.21	92.81 ±0.22	92.40 ±0.20	94.60 ±0.22	**93.99** ±0.21	94.01 ±0.20	94.12 ±0.20

Table 3.7: Experimental Results for Baseline Activations in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.6 given in the CIFAR Section.

Activation Function	ResNext	Squeezenet	AlexNet	LeNet	Xception	SeNet 18	SeNet 34	SeNet 50
ReLU	92.95 ±0.22	90.10 ±0.23	83.86 ±0.22	75.45 ±0.21	90.47 ±0.23	94.01 ±0.20	94.19 ±0.19	94.29 ±0.20
Leaky ReLU	93.10 ±0.21	90.22 ±0.24	83.81 ±0.20	75.55 ±0.23	90.62 ±0.25	94.10 ±0.21	94.20 ±0.17	94.18 ±0.20
PReLU	93.18 ±0.23	90.40 ±0.22	83.94 ±0.22	75.71 ±0.24	90.60 ±0.23	94.17 ±0.22	94.41 ±0.19	94.47 ±0.23
ReLU6	93.22 ±0.20	90.39 ±0.21	84.10 ±0.20	75.85 ±0.21	90.52 ±0.22	94.21 ±0.22	94.34 ±0.18	94.39 ±0.21
ELU	93.22 ±0.22	90.32 ±0.22	83.99 ±0.23	75.48 ±0.21	90.78 ±0.24	94.29 ±0.24	94.28 ±0.19	94.29 ±0.23
Softplus	92.79 ±0.25	90.01 ±0.26	83.65 ±0.26	75.32 ±0.28	90.49 ±0.29	94.14 ±0.25	94.11 ±0.22	93.91 ±0.24
PAU	93.72 ±0.18	90.64 ±0.20	84.71 ±0.19	76.10 ±0.21	90.98 ±0.20	94.74 ±0.23	94.89 ±0.20	94.90 ±0.21
Swish	93.64 ±0.20	90.89 ±0.21	84.85 ±0.20	76.45 ±0.19	90.81 ±0.18	94.61 ±0.22	94.81 ±0.23	94.97 ±0.23
GELU	93.87 ±0.22	90.71 ±0.20	85.10 ±0.22	76.59 ±0.23	90.97 ±0.20	94.70 ±0.21	94.89 ±0.23	95.10 ±0.24
SAU	**94.37** ±0.20	**91.42** ±0.19	**85.72** ±0.20	**77.01** ±0.20	**91.59** ±0.21	**95.21** ±0.19	**95.29** ±0.20	**95.57** ±0.24

Table 3.8: Experimental Results for Baseline Activations and SAU in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	Inception V3	WideRes Net 28-10	DenseNet 121	Effitient Net B0	VGG16	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50
Leaky ReLU	93.80 ±0.20	94.97 ±0.22	94.49 ±0.17	94.51 ±0.19	93.10 ±0.24	93.41 ±0.19	93.97 ±0.16	94.27 ±0.18
PReLU	93.87 ±0.22	94.85 ±0.22	94.57 ±0.19	94.67 ±0.20	93.19 ±0.25	93.31 ±0.21	91.21 ±0.19	94.39 ±0.21
ReLU6	93.97 ±0.19	95.14 ±0.20	94.71 ±0.19	94.40 ±0.20	93.21 ±0.22	93.62 ±0.20	93.82 ±0.17	94.51 ±0.16
ELU	93.85 ±0.20	95.05 ±0.22	94.79 ±0.21	94.57 ±0.22	93.15 ±0.20	93.78 ±0.23	93.70 ±0.18	94.50 ±0.17
Softplus	93.52 ±0.26	94.71 ±0.25	94.45 ±0.22	94.77 ±0.23	93.02 ±0.26	93.23 ±0.24	91.29 ±0.22	94.41 ±0.23
PAU	94.10 ±0.20	94.57 ±0.21	94.83 ±0.20	94.89 ±0.21	93.41 ±0.24	94.22 ±0.20	94.46 ±0.21	94.51 ±0.22
Swish	94.01 ±0.22	94.61 ±0.23	94.71 ±0.21	94.65 ±0.22	93.52 ±0.23	**94.65** ±0.22	94.58 ±0.23	94.67 ±0.20
GELU	94.12 ±0.21	94.50 ±0.22	94.95 ±0.22	94.61 ±0.20	**93.59** ±0.21	94.30 ±0.22	94.45 ±0.22	94.61 ±0.21

Table 3.9: Experimental Results for Baseline Activations in CIFAR10 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.6 given in the CIFAR Section.

Activation Function	ResNext	Squeezenet	AlexNet	LeNet	Xception	SeNet 18	SeNet 34	SeNet 50
ReLU	74.02 ±0.24	65.95 ±0.22	54.51 ±0.27	45.08 ±0.29	70.89 ±0.23	74.37 ±0.22	75.01 ±0.23	75.89 ±0.20
Leaky ReLU	74.32 ±0.26	66.21 ±0.23	54.89 ±0.24	45.10 ±0.29	71.35 ±0.25	74.67 ±0.23	75.18 ±0.23	76.20 ±0.22
PReLU	74.61 ±0.25	66.51 ±0.24	55.30 ±0.22	45.29 ±0.28	71.59 ±0.27	74.54 ±0.23	75.32 ±0.22	76.61 ±0.23
ReLU6	74.52 ±0.24	66.23 ±0.22	55.52 ±0.21	45.10 ±0.26	71.49 ±0.26	74.32 ±0.22	75.20 ±0.21	76.78 ±0.22
ELU	74.77 ±0.23	66.35 ±0.24	56.52 ±0.22	45.56 ±0.25	71.78 ±0.24	74.56 ±0.23	75.29 ±0.22	76.97 ±0.24
Softplus	73.89 ±0.27	66.10 ±0.26	54.45 ±0.28	45.56 ±0.29	70.77 ±0.25	74.07 ±0.24	74.78 ±0.25	75.98 ±0.22
PAU	75.86 ±0.23	66.78 ±0.20	57.89 ±0.27	46.75 ±0.27	73.10 ±0.24	74.52 ±0.22	75.18 ±0.20	77.39 ±0.18
Swish	75.36 ±0.22	66.42 ±0.22	57.32 ±0.26	46.54 ±0.26	72.77 ±0.22	74.45 ±0.23	75.58 ±0.21	77.10 ±0.17
GELU	75.52 ±0.22	66.69 ±0.22	57.56 ±0.28	46.45 ±0.29	72.97 ±0.25	74.47 ±0.24	75.47 ±0.22	77.14 ±0.21
SAU	**76.80** ±0.23	**68.01** ±0.19	**60.85** ±0.25	**47.10** ±0.26	**74.09** ±0.25	**75.64** ±0.20	**76.10** ±0.21	**78.64** ±0.18

Table 3.10: Experimental Results for Baseline Activations and SAU in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MobileNet V1	MobileNet V2	ResNet 18	ResNet 34	ResNet 50
Leaky ReLU	61.99 ±0.29	64.39 ±0.29	66.59 ±0.27	67.41 ±0.26	70.90 ±0.26	74.10 ±0.25	73.29 ±0.24	73.49 ±0.24	74.15 ±0.22
PReLU	62.20 ±0.27	64.12 ±0.28	66.84 ±0.29	67.65 ±0.25	71.10 ±0.26	74.19 ±0.27	73.39 ±0.25	73.61 ±0.23	74.41 ±0.23
ReLU6	62.12 ±0.26	64.32 ±0.26	66.72 ±0.26	67.52 ±0.24	71.32 ±0.24	74.28 ±0.25	73.20 ±0.25	73.13 ±0.22	74.40 ±0.21
ELU	62.10 ±0.27	64.52 ±0.26	66.51 ±0.28	67.62 ±0.24	71.20 ±0.25	74.35 ±0.26	73.23 ±0.22	73.52 ±0.23	74.29 ±0.23
Softplus	61.75 ±0.30	64.42 ±0.31	66.51 ±0.29	67.49 ±0.28	70.95 ±0.25	74.01 ±0.27	73.15 ±0.26	73.20 ±0.26	74.25 ±0.26
PAU	63.20 ±0.27	66.50 ±0.25	69.12 ±0.24	70.18 ±0.24	71.25 ±0.25	74.72 ±0.23	74.07 ±0.22	73.68 ±0.23	75.51 ±0.22
Swish	63.11 ±0.26	66.31 ±0.26	69.01 ±0.26	70.59 ±0.25	71.39 ±0.26	74.56 ±0.24	**74.54** ±0.21	74.10 ±0.22	75.45 ±0.23
GELU	63.35 ±0.25	66.10 ±0.25	69.39 ±0.26	70.79 ±0.25	71.14 ±0.26	74.68 ±0.25	74.18 ±0.24	73.87 ±0.24	75.30 ±0.23

Table 3.11: Experimental Results for Baseline Activations in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.5 given in the CIFAR Section.

Activation Function	Inception V3	WideRes Net 28-10	DenseNet 121	Effitient Net B0	VGG16	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50
Leaky ReLU	74.25 ±0.26	75.74 ±0.24	75.97 ±0.26	76.29 ±0.26	71.01 ±0.30	72.82 ±0.22	72.97 ±0.26	73.49 ±0.23
PReLU	74.37 ±0.27	75.91 ±0.26	76.04 ±0.27	76.41 ±0.28	71.16 ±0.31	72.80 ±0.26	73.40 ±0.25	73.87 ±0.24
ReLU6	74.24 ±0.24	75.98 ±0.24	75.91 ±0.23	76.11 ±0.25	71.10 ±0.29	72.59 ±0.25	73.12 ±0.23	73.61 ±0.23
ELU	74.37 ±0.25	76.10 ±0.26	75.80 ±0.24	76.01 ±0.26	71.05 ±0.30	72.25 ±0.28	73.35 ±0.26	74.10 ±0.25
Softplus	74.10 ±0.28	75.56 ±0.28	75.69 ±0.26	75.78 ±0.29	71.02 ±0.31	71.96 ±0.30	73.12 ±0.30	74.18 ±0.29
PAU	75.10 ±0.22	75.98 ±0.23	76.22 ±0.24	76.55 ±0.24	71.68 ±0.27	73.95 ±0.23	73.99 ±0.23	75.44 ±0.24
Swish	74.79 ±0.23	75.64 ±0.22	75.91 ±0.22	76.30 ±0.24	**71.92** ±0.25	73.72 ±0.25	74.35 ±0.24	75.57 ±0.23
GELU	74.64 ±0.23	76.18 ±0.25	76.31 ±0.22	76.97 ±0.24	71.52 ±0.26	**74.11** ±0.22	74.20 ±0.23	75.42 ±0.22

Table 3.12: Experimental Results for Baseline Activations in CIFAR100 Dataset for Image Classification Problem on Different Popular Network Architectures. We Report top-1 Test Accuracy (in %) for the Mean of 10 Different Runs. mean±std is Reported in the Table. This Table is an extension to the Table 3.5 given in the CIFAR Section.

for the mean of 10 different runs. We use the same experimental setup as used for the CIFAR100 dataset.

Activation Function	ShuffleNet V2 (2.0x)	ResNet 50	ResNet 18
ReLU	69.10 ± 0.24	75.10 ± 0.23	73.88 ± 0.24
Leaky ReLU	69.04 ± 0.23	75.04 ± 0.23	73.97 ± 0.26
PReLU	69.29 ± 0.25	75.17 ± 0.25	74.12 ± 0.25
ReLU6	69.36 ± 0.23	75.27 ± 0.22	74.17 ± 0.23
ELU	69.34 ± 0.24	75.32 ± 0.24	74.03 ± 0.24
Softplus	68.84 ± 0.28	74.52 ± 0.26	73.69 ± 0.27
Swish	72.78 ± 0.21	76.42 ± 0.22	74.39 ± 0.23
GELU	72.91 ± 0.22	76.54 ± 0.23	74.51 ± 0.23
PAU	73.09 ± 0.22	76.77 ± 0.22	74.62 ± 0.25
SAU	**74.22** ± 0.21	**77.81** ± 0.21	**75.59** ± 0.21

Table 3.13: Top-1 Test Accuracy Reported with Mixup Augmentation Method on CIFAR100 Dataset for the Mean of 10 Different Runs. mean±std is Reported in the Table

Tiny Imagenet:

This section presents results on the Tiny ImageNet dataset, a similar kind of image classification database like the ImageNet Large Scale Visual Recognition Challenge(ILSVRC). Tiny Imagenet contains 64×64 RGB images with total 100,000 training images, 10,000 validation images, and 10,000 test images and have total 200 image classes. We report the mean of 6 different runs for Top-1 accuracy in table 3.14 on WideResNet 28-10

(WRN 28-10) (Zagoruyko and Komodakis (2016)) and ResNet 18 (He *et al.* (2015*a*)) models. We consider a batch size of 64, 0.2 dropout rate (Srivastava *et al.* (2014)), SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)), He Normal initializer (He *et al.* (2015*b*)), initial learning rate(lr rate) 0.1, and lr rate is reduced by a factor of 10 after every 50 epochs up-to 300 epochs. Standard data augmentation techniques like rotation, width shift, height shift, shearing, zoom, horizontal flip, fill mode is applied to improve performance. It is evident from the table that the proposed function performs better than the baseline functions, and top-1 accuracy is stable (mean±std) and got a good improvement for SAU over ReLU.

Activation Function	WideResNet 28-10	ResNet 18
ReLU	62.77 ± 0.46	58.27 ± 0.42
Leaky ReLU	62.72 ± 0.46	58.52 ± 0.44
PReLU	62.70 ± 0.48	58.39 ± 0.44
ReLU6	62.59 ± 0.46	58.67 ± 0.41
ELU	62.58 ± 0.50	58.62 ± 0.43
Softplus	61.77 ± 0.59	58.04 ± 0.47
PAU	63.62 ± 0.44	59.47 ± 0.40
Swish	63.47 ± 0.46	59.02 ± 0.42
GELU	63.26 ± 0.48	59.27 ± 0.39
SAU	**64.07** ± 0.44	**60.12** ± 0.40

Table 3.14: A Detailed Comparison between SAU Activation and Other Baseline Activations in Tiny ImageNet Dataset for Image Classification Problem. We Report top-1 Test Accuracy (in %) for the Mean of 6 Different Runs. mean±std is Reported in the Table.

ImageNet-1k:

ImageNet-1k is a popular image database with more than 1.2 million training images and have 1000 classes. We report result on ImageNet-1k with ShuffleNet V2 (Ma *et al.* (2018)) and ResNet-50 He *et al.* (2015*a*) model in Table 3.15. We use a batch size of 256, SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)), 0.9 momentum, $5e^{-4}$ weight decay. We consider a linear decay learning rate scheduler from 0.1 and trained upto 600k iterations. Experiments on ImageNet-1k is conducted on four NVIDIA V100 GPUs with 32GB RAM each.

Activation Function	ShuffleNet V2 (1.0x)	ResNet-50
ReLU	69.31	75.50
Leaky ReLU	69.25	75.64
PReLU	69.20	75.48
ReLU6	69.44	75.77
ELU	69.62	75.54
Softplus	69.21	75.37
Swish	70.45	76.39
GELU	70.31	76.12
PAU	70.64	76.22
SAU	**71.52**	**77.47**

Table 3.15: top-1 Accuracy Reported on ImageNet-1k Dataset.

3.6.2 Object Detection

A standard problem in computer vision is object detection, in which the network model try to locate and identify each object present in the image. Object detection is widely used in face detection, autonomous vehicle etc. In this section, we present our results on challenging Pascal VOC dataset (Everingham *et al.* (2010)) on Single Shot Multi-Box Detector(SSD) 300 (Liu *et al.* (2016)) with VGG-16(with batch-normalization) (Simonyan and Zisserman (2015)) as the backbone network. No pre-trained weight is considered for our experiments in the network. The network has been trained with a batch size of 8, SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) with 0.9 momentum, $5e^{-4}$ weight decay, 0.001 learning rate, and trained up to 120000 iterations. We report the mean average precision (mAP) in Table 3.16 for the mean of 6 different runs.

Activation Function	mAP
ReLU	77.2±0.14
Leaky ReLU	77.2±0.19
PReLU	77.2±0.20
ReLU6	77.1±0.15
ELU	75.1±0.22
Softplus	74.2±0.25
PAU	77.4±0.14
Swish	77.3±0.11
GELU	77.3±0.12
SAU	**77.7**±0.10

Table 3.16: A Detailed Comparison between SAU Activation and Other Baseline Activations on Pascal VOC Dataset for Object Detection Problem with SSD300 Network Architecture. We Report mAP for the Mean of 6 Different Runs. mean±std is Reported in the Table.

3.6.3 Semantic Segmentation

Semantic segmentation is a computer vision problem that narrates the procedure of associating each pixel of an image with a class label. We present our experimental results in this section on the popular Cityscapes dataset (Cordts *et al.* (2016)). The U-net model (Ronneberger *et al.* (2015)) is considered as the segmentation framework and is trained up-to 250 epochs, with adam optimizer (Kingma and Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)). We report the mean of 6 different runs for Pixel Accuracy and the mean Intersection-Over-Union (mIOU) on test data on table 3.17.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.45±0.47	69.39±0.28
PReLU	78.88±0.40	68.80±0.40
ReLU6	79.67±0.40	69.79±0.42
Leaky ReLU	79.32±0.40	69.60±0.40
ELU	79.38±0.51	68.10±0.40
Softplus	78.60±0.49	68.20±0.49
PAU	79.52±0.49	69.12±0.31
Swish	79.99±0.47	69.61±0.29
GELU	80.10±0.37	69.39±0.38
SAU	**81.11**±0.40	**71.02**±0.32

Table 3.17: A Detailed Comparison between SAU Activation and Other Baseline Activations in CityScapes Dataset for Semantic Segmentation Problem on U-NET Model. We Report Pixel Accuracy and mIOU for the Mean of 6 Different Runs. mean±std is Reported in the Table.

3.6.4 Machine Translation

Machine Translation is a deep learning technique in which a model translate text or speech from one language to another language. In this section, we report results on WMT 2014 English→German dataset. The database contains 4.5 million training sentences. Network performance is evaluated on the newstest2014 dataset using the BLEU score metric. An Attention-based 8-head transformer network (Vaswani *et al.* (2017)) in trained with Adam optimizer (Kingma and Ba (2015)), 0.1 dropout rate (Srivastava *et al.* (2014)), and trained up to 100000 steps. Other hyperparameters are kept similar as mentioned in the original paper (Vaswani *et al.* (2017)). We report the mean of 6

different runs on Table 3.18 on the test dataset(newstest2014).

Activation Function	BLEU Score on the newstest2014 dataset
ReLU	26.2±0.15
Leaky ReLU	26.3±0.17
PReLU	26.2±0.21
ReLU6	26.1±0.14
ELU	25.1±0.15
Softplus	23.6±0.16
PAU	26.3±0.14
Swish	26.4±0.10
GELU	26.4±0.19
SAU	**26.7**±0.12

Table 3.18: A Detailed Comparison between SAU Activation and Other Baseline Activations in WMT-2014 Dataset for Machine Translation Problem on Transformer Model. We Report BLEU Score for the Mean of 6 Different Runs. mean±std is Reported in the Table.

3.7 Baseline Table

In this section, we present a table for SAU and the other baseline functions, which shows that SAU beat or perform equally well compared to baseline activation functions in most cases. We report a detailed comparison with SAU and the baseline activation functions based on all the experiments in earlier sections in Table 3.19. We notice that SAU performs remarkably well in most of the cases when compared with the baseline activations.

Baselines	ReLU	Leaky ReLU	PReLU	ReLU6	ELU	Softplus	PAU	Swish	GELU
SAU $>$ Baseline	71	71	71	71	71	72	67	66	67
SAU $=$ Baseline	0	0	0	0	0	0	0	0	0
SAU $<$ Baseline	1	1	1	1	1	0	5	6	5

Table 3.19: Baseline Table for SAU. In the Table, We Report the Total Number of Cases in Which SAU Underperforms, Equal, or Outperforms When We Compare with the Baseline Activation Functions

3.8 Computational Time Comparison

HP-1, HP-2 contains trainable parameters, which increases the complexity of the network, and due to this, there is a trade-off between network performance and computational cost. We have reported the Computational time comparison for HP-1, HP-2, and the baseline activation functions for both forward and backward pass on a 32×32 RGB image on ResNet-18 model in Table 3.20 for the mean of 100 runs. We have used an NVIDIA Tesla V100 GPU with 32GB ram.

Activation Function	Forward Pass	Backward Pass
ReLU	6.51 ± 0.35 μs	6.42 ± 0.81 μs
Leaky ReLU	6.61 ± 0.40 μs	6.52 ± 0.89 μs
PReLU	$8.64 \pm 1.50\mu$s	9.52 ± 1.75 μs
ReLU6	6.58 ± 0.50 μs	6.49 ± 0.85 μs
ELU	6.49 ± 0.52 μs	6.56 ± 0.81 μs
Softplus	6.59 ± 0.49 μs	6.38 ± 0.50 μs
GELU	10.91 ± 1.59 μs	12.62 ± 1.70 μs
Swish	10.59 ± 1.19 μs	12.60 ± 1.33 μs
PAU	18.69 ± 3.21 μs	25.91 ± 5.21 μs
SAU	12.96 ± 2.00 μs	$17.34 \pm 1.31\mu$s

Table 3.20: Runtime comparison for the forward and backward passes for SAU and other baseline activation functions for a $32\times$ 32 RGB image in ResNet-18 model.

3.9 Conclusion

In this chapter, a new novel smooth activation function using approximate identity has been proposed, and the proposed function is called Smooth Activation Unit (SAU). The proposed function can approximate ReLU or its different variants (like Leaky ReLU etc.) quite well. For all experiments, SAU is considered as a trainable activation function. It has been shown that in a wide range of experiments on different deep learning problems, the proposed functions outperform the known activations like ReLU, Leaky ReLU or Swish in most cases which shows that replacing the hand-crafted activation functions by SAU can be beneficial in deep networks.

Though SAU improves network performance in different deep learning problems, it is slower than other smooth activation functions like Swish, GELU etc. (but still faster than PAU). To address this drawback of SAU, two better activations have been proposed in the next chapter using an approximation by a smooth function of the maximum function.

CHAPTER 4

Smooth Maximum Unit[1]

4.1 Introduction

This chapter proposes two smooth novel activations based on the approximation by a smooth function of the maximum function. Deep Neural network has emerged significantly in recent years and impacted our real-life applications. Neural networks are the backbone of deep learning. An activation function is the brain of the neural network, which plays a central role in the effectiveness & training dynamics of deep neural networks. Hand-designed activation functions are quite a common choice in neural network models. ReLU (Nair and Hinton (2010)) is a widely used hand-designed activation function. Despite its simplicity, ReLU has a major drawback, known as the dying ReLU problem, in which up to 50% neurons can be dead during network training. To overcome the shortcomings of ReLU, many activations have been proposed in recent years. Leaky ReLU (Maas *et al.* (2013*a*)), Parametric ReLU (He *et al.* (2015*b*)), ELU (Clevert *et al.* (2016)), Softplus (Zheng *et al.* (2015)), Randomized Leaky ReLU (Xu *et al.* (2015*a*)) are a few of them though they marginally improve performance of ReLU. Swish (Ramachandran *et al.* (2017)) is a non-linear activation function proposed by the Google brain team, showing some good improvement of ReLU. GELU (Hendrycks and Gimpel (2020)) is another popular smooth activation function. It can be shown that Swish and GELU are both approximation function of ReLU. Recently, a few non-linear activations have been proposed that improve the performance of ReLU, Swish or GELU. Some of them are either hand-designed or approximation by a smooth function of the Leaky ReLU function. Mish (Misra (2020)) and Padé activation unit (Molina *et al.* (2020)) are a few of them.

In the previous chapter, a novel activation called SAU was presented. Though SAU improves performance compared to the widely used activation functions, SAU has a

[1]This chapter is a slightly modified version of the paper published in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Biswas *et al.* (2022).

drawback. SAU takes higher training time than other smooth activations like Swish, GELU etc. In this chapter, the problem has been tried to resolve. The proposed activation functions (SMU & SMU-1) have either similar or lesser training time compared to Swish, GELU etc., while both SMU and SMU-1 perform better than the earlier mentioned activation functions.

4.2 Related Works and Motivation

In a deep neural network, activations are either fixed before training or trainable. Researchers have proposed several activations in recent years by combining known functions. Some of these functions have hyperparameters or trainable parameters. In the case of trainable activation functions, parameters are optimized during training. Swish is a popular activation function that can be used as a constant or trainable activation function. It performs well in various deep learning tasks like image classification, object detection, machine translation etc. GELU shares similar properties like the Swish activation function, and it has gained popularity in the deep learning community due to its efficacy in natural language processing tasks. GELU has been used in BERT (Devlin *et al.* (2018)), GPT-2 (Radford *et al.* (2019)), and GPT-3 (Brown *et al.* (2020)) architectures. Padé activation unit (PAU) has been proposed recently, and it is constructed from the approximation of the Leaky ReLU function by rational polynomials of a given order. Though PAU improves network performance in the image classification problem over ReLU, its variants, and Swish, it has a major drawback. PAU contains many trainable parameters, and due to this, PAU significantly increases the network complexity and computational cost. The proposed method in this chapter tried to solve this problem.

4.3 Research contribution

In this chapter, we propose activation functions using the smoothing maximum technique. The maximum function is non-smooth at the origin. We want to explore how does the approximation by a smooth function of the maximum function (which can be used as an activation function) affects a network's training dynamics and performance. Our experimental evaluation shows that our proposed activation functions are compar-

atively more effective than ReLU, Mish, Swish, GELU, PAU etc., across different deep learning tasks. This chapter can be summarised as follows:

1. We have proposed activation functions by smoothing the maximum function. We show that it can approximate GELU, ReLU, Leaky ReLU or the general Maxout family.
2. We show that the proposed functions outperform widely used activation functions in a variety of deep learning tasks.

4.4 Smooth Maximum Unit

We use approximation by a smooth function of the maximum function to construct a smooth activation function. We refer to this function as the Smooth Maximum Unit (SMU). Using the approximation by a smooth function of the $|x|$ function, one can find a general approximating formula for the maximum function, which can smoothly approximate the general Maxout (Goodfellow *et al.* (2013)) family, ReLU, Leaky ReLU or its variants, Swish etc. We also show that the well established GELU (Hendrycks and Gimpel (2020)) function can be obtained as a special case of SMU.

4.4.1 Approximation by a smooth function of the maximum function

Note that the maximum function can be expressed as following two different ways:

$$\begin{aligned} max(x_1, x_2) &= \begin{cases} x_1 & \text{if } x_1 \geq x_2 \\ x_2 & \text{otherwise} \end{cases} \\ &= \frac{(x_1 + x_2) + |x_1 - x_2|}{2} \end{aligned} \tag{4.1}$$

Note that the max function is not differentiable at the origin. Using approximations of the $|x|$ function by a smooth function, we can create approximations to the maximum functions. There are many known approximations to $|x|$, but for the rest of this article, we will focus on two specific approximations of $|x|$, namely $x\text{erf}(\mu x)$ and $\sqrt{x^2 + \mu^2}$. We noticed that the activations constructed using these two functions provide good performance on standard datasets on different deep learning problems. Note

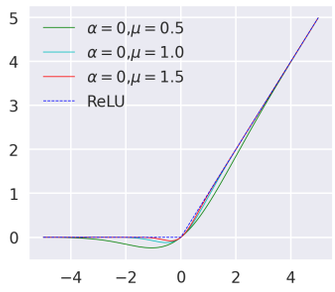

Figure 4.1: Approximation of ReLU using SMU ($\alpha = 0$) for different values of μ. As $\mu \to \infty$, SMU smoothly approximate ReLU

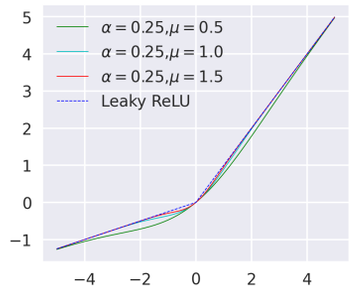

Figure 4.2: Approximation of Leaky ReLU ($\alpha = 0.25$) using SMU for different values of μ. As $\mu \to \infty$, SMU smoothly approximate Leaky ReLU

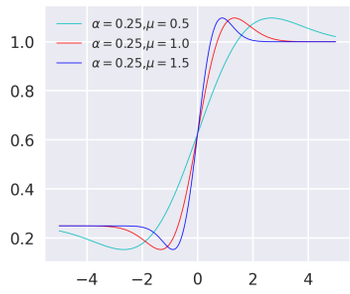

Figure 4.3: First order derivatives of SMU for $\alpha = 0.25$ and different values of μ.

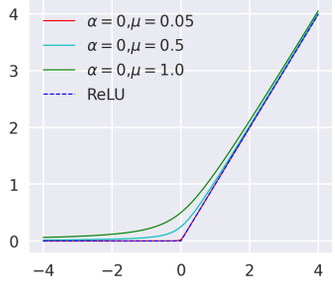

Figure 4.4: Approximation of ReLU using SMU-1 ($\alpha = 0$) for different values of μ. As $\mu \to 0$, SMU-1 smoothly approximate ReLU

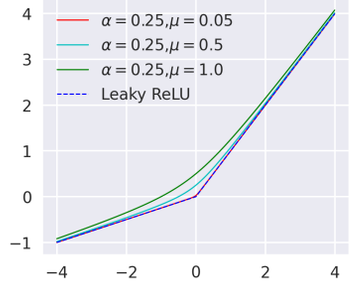

Figure 4.5: Approximation of Leaky ReLU ($\alpha = 0.25$) using SMU-1 for different values of μ. As $\mu \to 0$, SMU-1 smoothly approximate Leaky ReLU

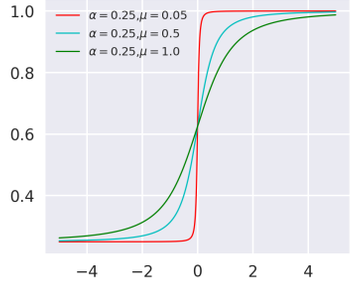

Figure 4.6: First order derivatives of SMU-1 for $\alpha = 0.25$ and different values of μ.

that $\sqrt{x^2 + \mu^2}$ as $\mu \to 0$ approximate $|x|$ from above while $x\text{erf}(\mu x)$. as $\mu \to \infty$ gives an approximation of $|x|$ from below. Here erf is the Gaussian error function defined as follows:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt.$$

Now, replacing the $|x|$ function by $x\text{erf}(\mu x)$ in equation (4.1), we have the approxi-

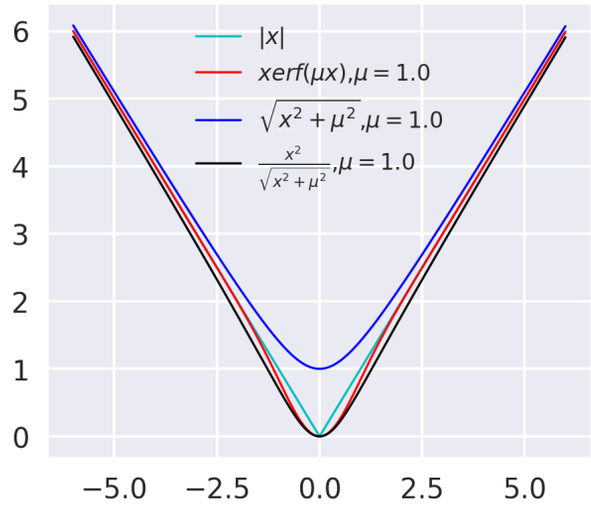

Figure 4.7: Approximation by a smooth function of $|x|$

mation by a smooth function formula for the maximum function as follows:

$$f_1(x_1, x_2; \mu) = \frac{(x_1 + x_2) + (x_1 - x_2)\ \text{erf}(\mu(x_1 - x_2))}{2}. \tag{4.2}$$

Similarly, we can derive the the approximation by a smooth function formula for the maximum function from equation (4.1) by replacing the $|x|$ function by $\sqrt{x^2 + \mu^2}$ as follows:

$$f_2(x_1, x_2; \mu) = \frac{(x_1 + x_2) + \sqrt{(x_1 - x_2)^2 + \mu^2}}{2} \tag{4.3}$$

Note that as $\mu \to \infty$, $f_1(x_1, x_2; \mu) \to \max(x_1, x_2)$ and as $\mu \to 0$, $f_2(x_1, x_2; \mu) \to \max(x_1, x_2)$. For particular values of x_1 and x_2, we can approximate known activation functions. For example, consider $x_1 = ax$, $x_2 = bx$, with $a \neq b$ in (4.2), we get:

$$f_1(ax, bx; \mu) = \frac{(a + b)x + (a - b)x\ \text{erf}(\mu(a - b)x)}{2}. \tag{4.4}$$

This is a simple case from the Maxout family (Goodfellow *et al.* (2013)) while more complicated cases can be found by considering nonlinear choices of x_1 and x_2. We can similarly get approximation by a smooth function formula to ReLU and Leaky ReLU. For example, consider $x_1 = x$ and $x_2 = 0$, we have approximation by a smooth function

of ReLU as follows:

$$f_1(x, 0; \mu) = \frac{x + x \text{ erf}(\mu x)}{2}. \tag{4.5}$$

We know that GELU (Hendrycks and Gimpel (2020)) is a approximation by a smooth function of ReLU. Notice that, if we choose $\mu = \frac{1}{\sqrt{2}}$ in equation (4.5), we can recover GELU activation function which also show that GELU is approximation by a smooth function of ReLU. Also, considering $x_1 = x$ and $x_2 = \alpha x$, we have a approximation by a smooth function of Leaky ReLU or Parametric ReLU depending on whether α is a hyperparameter or a learnable parameter.

$$f_1(x, \alpha x; \mu) = \frac{(1 + \alpha)x + (1 - \alpha)x \text{ erf}(\mu(1 - \alpha)x)}{2}. \tag{4.6}$$

Note that, equation (4.5) and equation (4.6) approximate ReLU or Leaky ReLU from below. Similarly, we can derive approximating function from equation (4.3) which will approximate ReLU or Leaky ReLU from above.

The corresponding derivatives of equation (4.6) for input variable x is

$$\begin{aligned} \frac{d}{dx} f_1(x, \alpha x; \mu) = \frac{1}{2}[(1 + \alpha) + (1 - \alpha) \text{ erf}(\mu(1 - \alpha)x) \\ + \frac{2}{\sqrt{\pi}} \mu(1 - \alpha)^2 x e^{-(\mu(1-\alpha)x)^2}] \end{aligned} \tag{4.7}$$

$$\text{where } \frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}} e^{-x^2}.$$

Figures 4.1, 4.2, and 4.3 show the plots for $f_1(x, 0; \mu)$, $f_1(x, 0.25x; \mu)$, and derivative of $f_1(x, 0.25x; \mu)$ for different values of μ. From the figures it is clear that as $\mu \to \infty$, $f_1(x, \alpha x; \mu)$ smoothly approximate ReLU or Leaky ReLU depending on value of α. We call the function in equation (4.6) as Smooth Maximum Unit (SMU). Similarly, We can derive a function by replacing $x_1 = x$ and $x_2 = \alpha x$ in equation (4.3) and we call this function SMU-1. For all of our experiments, we will use SMU and SMU-1 as our proposed activation functions.

Figure 4.4, 4.5 represents approximation of ReLU ($\alpha = 0$), Leaky ReLU ($\alpha = 0.25$) by SMU-1 for different values of μ and as $\mu \to 0$, SMU-1 overlap ReLU or Leaky ReLU depending on the value of α. Figure 4.6 represents the derivatives of SMU-1 for $\alpha = 0.25$ and different valus of μ.

There are many known approximation by a smooth function to the $|x|$ function like $x\,\text{erf}(\mu x)$, $\sqrt{x^2+\mu^2}$, $\frac{x^2}{\sqrt{x^2+\mu^2}}$ etc. As $\mu \to 0$, $\frac{x^2}{\sqrt{x^2+\mu^2}}$ gives approximation by a smooth function of $|x|$ from below. We give a plot of well known approximation to $|x|$ in Figure 4.7.

Replace $x_1 = x$ and $x_2 = \alpha x$ in equation (3), we have a approximation by a smooth function of Leaky ReLU or Parametric ReLU depending on whether α is a hyperparameter or a learnable parameter. We call it SMU-1 and is defined as

$$f_2(x, \alpha x; \mu) = \frac{(1+\alpha)x + \sqrt{(1-\alpha)^2x^2+\mu^2}}{2}$$

and the corresponding derivative with respect to input variable x is

$$\frac{d}{dx} f_2(x, \alpha x; \mu) = \frac{(1+\alpha) + \frac{(1-\alpha)^2 x}{\sqrt{(1-\alpha)^2x^2+\mu^2}}}{2}$$

.

4.4.2 Learning activation parameters via back-propagation

Trainable activation function parameters are updated using backpropagation (LeCun *et al.* (1989)) technique (see He *et al.* (2015*b*)) according to (4.8) and for a single layer, the gradient of a hyper-parameter ω is:

$$\frac{\partial L}{\partial \omega} = \sum_x \frac{\partial L}{\partial f(x)} \frac{\partial f(x)}{\partial \omega} \tag{4.8}$$

where L is the objective function, $\omega \in \{\alpha, \mu\}$ and $f(x) \in \{f_1(x, \alpha x; \mu), f_2(x, \alpha x; \mu)\}$. We implemented forward pass in both Pytorch (Paszke *et al.* (2019)) & Tensorflow-Keras (Chollet *et al.* (2015)) API, and automatic differentiation will update the parameters. Alternatively, CUDA (Nickolls *et al.* (2008)) based implementation (see Maas *et al.* (2013*a*)) can be used and the gradients are given in (4.9) and (4.10)) for the parameters α and μ of equation (4.6) are as follows:

$$\frac{\partial f_1}{\partial \alpha} = \frac{x}{2} - \frac{x\,\text{erf}(\mu(1-\alpha)x)}{2} - \frac{(1-\alpha)\mu x^2 e^{-(\mu(1-\alpha)x)^2}}{\sqrt{\pi}} \tag{4.9}$$

$$\frac{\partial f_1}{\partial \mu} = \frac{1}{\sqrt{\pi}}(1-\alpha)^2 x^2 e^{-(\mu(1-\alpha)x)^2} \tag{4.10}$$

α and μ can be either hyperparameters or trainable parameters.

Now, note that the class of neural networks with SMU and SMU-1 activation functions are dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K.

The proof follows from the following proposition (see Molina *et al.* (2020)).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 Kidger and Lyons (2020)) :- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

4.5 Experiments

We report a detailed experimental evaluation in the next subsections on four different deep learning problems like image classification, object detection, semantic segmentation, and machine translation. To compare performance of our proposed activation function, we consider ten popular activation functions as the baseline functions. The following activations are considered to compare performance with SMU and SMU-1: ReLU (Nair and Hinton (2010)), Leaky ReLU (Maas *et al.* (2013*a*)), ReLU6 (Krizhevsky (2010)), Parametric ReLU (PReLU) (He *et al.* (2015*b*)), ELU (Clevert *et al.* (2016)), Softplus (Zheng *et al.* (2015)), Swish (Ramachandran *et al.* (2017)), Mish (Misra (2020)), GELU (Hendrycks and Gimpel (2020)), and Pade Activation Unit (PAU) (Molina *et al.* (2020)). For all experiments, we consider Swish ($x.\,\mathrm{Sigmoid}(\beta x)$), PReLU ($\max(\mathrm{x}, \mathrm{ax})$), and PAU as trainable activation functions. We initialize the trainable parameter β at 1.0 for Swish, a at 0.25 for PReLU. PAU function has ten trainable parameters and all the parameters are initialized as suggested in (Molina *et al.* (2020)). All the trainable parameters are updated via the backpropagation (LeCun *et al.* (1989)) algorithm. We report results for baseline activation functions, SMU and SMU-1 activation functions in the following sections. SMU-1 is a computationally cheap activation function due to its simple form, while it boosts the network performance remarkably

well in all the experiments compared to the baseline activations. All the experiments are conducted on an NVIDIA Tesla V100 GPU with 32GB RAM.

4.5.1 Image Classification

We report results for the image classification problem on six popular benchmarking datasets: MNIST, Fashion MNIST, SVHN, CIFAR10, CIFAR100, and Tiny ImageNet. Detailed results are reported in the following subsections. For SMU, we consider $\alpha = 0.25$, a constant hyperparameter and μ as a trainable parameter and initialise at 1.0.

MNIST, Fashion MNIST, and SVHN

In this section, We present our experimental comparison for SMU, SMU-1 and other baseline activations on MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), and SVHN (Netzer *et al.* (2011)) datasets. The MNIST and Fashion MNIST databases contain 60k training and 10k testing 28×28 grey-scale images. Both the datasets have ten different classes. The SVHN database has 32×32 RGB images and a total of 73257 training images and 26032 testing images with ten different classes. Standard data augmentation methods like zoom, rotation, height shift, shearing are applied to these three datasets. We consider a batch size of 128, 0.01 initial learning rate and decay the learning rate with cosine annealing (Loshchilov and Hutter (2017)) learning rate scheduler. We use stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 100 epochs. We report results with VGG-16 (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))) architecture in Table 4.1 for mean of 15 different runs. We report more results on MNIST, Fashion MNIST, and SVHN datasets with SMU, SMU-1 and baseline activations with LeNet, AlexNet, and a custom-designed model in Table 4.2, Table 4.3, and Table 4.4 respectively. Our custom homogeneous convolutional neural network has max-pooling layers(thrice), channel depths of size 128 (twice), 64 (thrice), 32 (twice), and a dense layer of size 128. Batch-normalization is applied before the activation function layer. We use 3×3 kernels in CNN layers and 2×2 kernels in max-pooling layers.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.53 ± 0.07	93.79 ± 0.15	95.97 ± 0.14
Leaky ReLU	99.58 ± 0.08	93.80 ± 0.15	96.02 ± 0.15
PReLU	99.55 ± 0.07	93.90 ± 0.17	96.10 ± 0.16
ReLU6	99.59 ± 0.06	93.93 ± 0.12	96.11 ± 0.15
ELU	99.48 ± 0.05	93.87 ± 0.16	96.05 ± 0.17
Softplus	99.22 ± 0.14	93.58 ± 0.18	95.81 ± 0.21
Swish	99.57 ± 0.05	94.17 ± 0.11	96.20 ± 0.12
Mish	99.63 ± 0.04	94.25 ± 0.13	96.31 ± 0.12
GELU	99.59 ± 0.04	94.22 ± 0.14	96.21 ± 0.14
PAU	99.55 ± 0.07	94.09 ± 0.14	96.20 ± 0.14
SMU	$\mathbf{99.69} \pm 0.04$	$\mathbf{94.48} \pm 0.10$	$\mathbf{96.59} \pm 0.11$
SMU-1	$\mathbf{99.65} \pm 0.04$	$\mathbf{94.37} \pm 0.14$	$\mathbf{96.43} \pm 0.14$

Table 4.1: Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on VGG16 architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.16 ± 0.11	91.42 ± 0.18	92.10 ± 0.21
Leaky ReLU	99.12 ± 0.12	91.43 ± 0.22	92.27 ± 0.20
ReLU6	99.21 ± 0.10	91.47 ± 0.19	92.28 ± 0.16
PReLU	99.23 ± 0.10	91.40 ± 0.20	92.09 ± 0.20
ELU	99.30 ± 0.10	91.41 ± 0.21	92.28 ± 0.19
Softplus	99.01 ± 0.19	91.11 ± 0.25	91.92 ± 0.26
GELU	99.33 ± 0.08	91.60 ± 0.13	92.47 ± 0.17
Swish	99.29 ± 0.09	91.66 ± 0.15	92.35 ± 0.20
PAU	99.37 ± 0.10	91.56 ± 0.14	92.37 ± 0.21
Mish	99.36 ± 0.06	$\mathbf{91.68} \pm 0.13$	92.41 ± 0.17
SMU	$\mathbf{99.47} \pm 0.04$	91.58 ± 0.16	$\mathbf{92.79} \pm 0.16$
SMU-1	$\mathbf{99.41} \pm 0.05$	91.51 ± 0.14	$\mathbf{92.66} \pm 0.17$

Table 4.2: Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on LeNet architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.48 ± 0.07	92.70 ± 0.20	95.03 ± 0.16
Leaky ReLU	99.47 ± 0.07	92.81 ± 0.19	95.10 ± 0.18
ReLU6	99.52 ± 0.05	92.94 ± 0.14	95.16 ± 0.14
PReLU	99.45 ± 0.09	92.79 ± 0.20	95.12 ± 0.16
ELU	99.51 ± 0.06	92.96 ± 0.15	95.19 ± 0.16
Softplus	99.27 ± 0.11	92.30 ± 0.27	94.71 ± 0.20
GELU	99.57 ± 0.07	93.09 ± 0.12	95.20 ± 0.13
Swish	99.59 ± 0.06	92.90 ± 0.17	95.35 ± 0.16
PAU	99.51 ± 0.10	93.06 ± 0.18	95.29 ± 0.15
Mish	99.61 ± 0.06	93.12 ± 0.15	95.31 ± 0.12
SMU	$\mathbf{99.68} \pm 0.04$	$\mathbf{93.31} \pm 0.15$	$\mathbf{95.59} \pm 0.12$
SMU-1	$\mathbf{99.65} \pm 0.05$	$\mathbf{93.20} \pm 0.11$	$\mathbf{95.46} \pm 0.13$

Table 4.3: Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on AlexNet architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.41 ± 0.09	92.97 ± 0.16	94.48 ± 0.14
Leaky ReLU	99.40 ± 0.07	93.17 ± 0.14	94.59 ± 0.18
ReLU6	99.46 ± 0.07	93.35 ± 0.16	94.61 ± 0.12
PReLU	99.37 ± 0.11	93.18 ± 0.13	94.58 ± 0.15
ELU	99.49 ± 0.07	93.24 ± 0.14	94.57 ± 0.15
Softplus	99.21 ± 0.14	92.99 ± 0.24	94.34 ± 0.22
GELU	99.52 ± 0.05	93.39 ± 0.14	94.87 ± 0.10
Swish	99.54 ± 0.07	93.34 ± 0.15	94.84 ± 0.14
PAU	99.55 ± 0.12	93.37 ± 0.17	94.79 ± 0.14
Mish	$\mathbf{99.64} \pm 0.06$	93.43 ± 0.12	94.87 ± 0.10
SMU	99.61 ± 0.06	$\mathbf{93.61} \pm 0.09$	$\mathbf{95.06} \pm 0.10$
SMU-1	99.57 ± 0.07	$\mathbf{93.49} \pm 0.11$	$\mathbf{95.18} \pm 0.12$

Table 4.4: Comparison between SMU, SMU-1 activations and other baseline activations on MNIST, Fashion MNIST, and SVHN datasets for image classification problem on custom designed architecture. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

CIFAR

In this section, we report results on the popular image classification datasets CIFAR10 (Krizhevsky (2009)) and CIFAR100 (Krizhevsky (2009)). Both the datasets have 50k training and 10k testing images. While CIFAR10 has ten classes and CIFAR100 has 100 classes. In these two datasets for all experiments, we consider a batch size of 128, 0.01 initial learning rate and decay the learning rate with cosine annealing (Loshchilov and Hutter (2017)) learning rate scheduler, stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. We consider standard data augmentation methods like horizontal flip and rotation. Top-1 accuracy is reported in Table 4.5, Table 4.7, Table 4.8, & Table 4.9 on CIFAR100 (Krizhevsky (2009)) dataset and Table 4.6, Table 4.10, Table 4.11 & Table 4.12 on CIFAR10 (Krizhevsky (2009)) dataset for mean of 15 different runs. The results are reported with MobileNet V1 (Howard *et al.* (2017)), MobileNet V2 (Sandler *et al.* (2019)), ShuffleNet V1 (Zhang *et al.* (2017*b*)) (SF V1), ShuffleNet V2 (Ma *et al.* (2018)), PreActResNet (He *et al.* (2016)), ResNet (He *et al.* (2015*a*)), GoogleNet (Szegedy *et al.* (2014*a*)), Inception V3 (Szegedy *et al.* (2015*a*)), DenseNet (Huang *et al.* (2016*a*)), Squeeze-and-Excitation Networks (SeNet) (Hu *et al.* (2017)), SqueezeNet (Iandola *et al.* (2016)), ResNext (Xie *et al.* (2017)), WideResNet (Zagoruyko and Komodakis (2016)), Xception (Chollet (2017)), VGG (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))), AlexNet (Krizhevsky *et al.* (2012)), LeNet (Lecun *et al.* (1998)), and EfficientNet B0 (Tan and Le (2020)). From Table 4.5 it is clear that Top-1 classification accuracy improves by 6.19%, 6.22%, 3.39%, 3.51%, 3.09%, 3.40% and 3.08% when we replace ReLU by SMU on the CIFAR100 dataset with ShuffleNet V2 (1.0x), ShuffleNet V2 (2.0x), PreActResNet-50, ResNet-50, ResNext, Xception and SeNet-50 models respectively. The Figures 4.8 and 4.9 shows the learning curves on CIFAR100 dataset with ShuffleNet V2 (2.0x) model for the baseline and the proposed activation functions.

Tiny Imagenet

In this section, We report results for classification problem on a more challenging dataset, Tiny Imagenet (Le and Yang (2015)). Tiny imagenet has RGB images of size

Model	ReLU	SMU	SMU-1
	Top-1 accuracy	Top-1 accuracy	Top-1 accuracy
Shufflenet V2 0.5x	62.07 ± 0.26	**66.67** ± 0.24	**65.60** ± 0.24
Shufflenet V2 1.0x	64.41 ± 0.25	**70.60** ± 0.21	**69.96** ± 0.22
Shufflenet V2 1.5x	67.20 ± 0.26	**72.68** ± 0.19	**72.05** ± 0.20
Shufflenet V2 2.0x	67.52 ± 0.25	**73.74** ± 0.20	**73.45** ± 0.23
PreActResNet 18	73.18 ± 0.22	**76.07** ± 0.20	**75.72** ± 0.22
PreActResNet 34	73.41 ± 0.24	**76.21** ± 0.20	**75.87** ± 0.21
PreActResNet 50	73.89 ± 0.23	**77.28** ± 0.17	**76.85** ± 0.20
ResNet 18	73.23 ± 0.26	**75.22** ± 0.20	**74.91** ± 0.20
ResNet 34	73.33 ± 0.27	**75.77** ± 0.20	**75.59** ± 0.21
ResNet 50	74.12 ± 0.24	**77.63** ± 0.20	**76.89** ± 0.23
SeNet 18	74.77 ± 0.22	**76.17** ± 0.17	**75.44** ± 0.20
SeNet 34	75.12 ± 0.22	**76.79** ± 0.18	**75.79** ± 0.21
SeNet 50	76.09 ± 0.20	**79.17** ± 0.16	**78.45** ± 0.20
ResNext	74.43 ± 0.22	**77.52** ± 0.18	**77.03** ± 0.21
MobileNet V1	71.10 ± 0.26	**73.59** ± 0.22	**73.10** ± 0.22
MobileNet V2	74.17 ± 0.24	**76.31** ± 0.19	**76.03** ± 0.19
Xception	71.22 ± 0.26	**74.62** ± 0.23	**74.11** ± 0.23
EffitientNet B0	76.60 ± 0.27	**79.10** ± 0.22	**78.77** ± 0.23

Table 4.5: Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

64×64 with total 1,00,000 training images, 10,000 validation images, and 10,000 test images with total 200 classes. Standard data augmentation methods like rotation, horizontal flip is applied. We consider a batch size of 64, 0.1 initial learning rate and reduce the learning rate after every 50 epochs by a factor of 10. We use stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 200 epochs. Results are reported with WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), DenseNet-121 (Huang *et al.* (2016*a*)), ResNet-18, and ResNet-50 (He *et al.* (2015*a*)) models and Top-1 classification accuracy is reported in table 4.14 for mean of 10 different runs. The proposed functions performs better than the baseline functions and results are stable (mean±std) and we get very good improvement over the baseline activation functions. Replacing ReLU by SMU, we have 2.56%, 2.23%, 2.31%, and 2.78% boost in Top-1 classification accuracy on DenseNet-121, ResNet-18, ResNet-50,

Model	ReLU	SMU	SMU-1
	Top-1 accuracy	Top-1 accuracy	Top-1 accuracy
ShuffleNet V2 0.5x	88.40 ± 0.22	**90.63** ± 0.16	**90.39** ± 0.18
ShuffleNet V2 1.0x	90.81 ± 0.24	**92.72** ± 0.18	**92.42** ± 0.20
ShuffleNet V2 1.5x	91.21 ± 0.22	**93.42** ± 0.17	**92.27** ± 0.18
ShuffleNet V2 2.0x	91.70 ± 0.20	**93.61** ± 0.14	**93.40** ± 0.16
PreActResNet 18	93.57 ± 0.20	**94.63** ± 0.15	**94.52** ± 0.17
PreActResNet 34	94.21 ± 0.17	**95.12** ± 0.13	**94.93** ± 0.14
PreActResNet 50	94.30 ± 0.18	**95.37** ± 0.11	**94.94** ± 0.12
ResNet 18	94.10 ± 0.20	**94.78** ± 0.17	**94.51** ± 0.19
ResNet 34	94.22 ± 0.18	**94.91** ± 0.16	**94.77** ± 0.17
ResNet 50	94.26 ± 0.18	**95.38** ± 0.16	**94.92** ± 0.17
SeNet 18	94.29 ± 0.20	**94.75** ± 0.17	**94.56** ± 0.19
SeNet 34	94.42 ± 0.20	**95.27** ± 0.15	**94.89** ± 0.17
SeNet 50	94.55 ± 0.19	**95.92** ± 0.12	**95.22** ± 0.17
ResNext	93.37 ± 0.18	**94.52** ± 0.15	**94.04** ± 0.18
MobileNet V1	92.41 ± 0.14	**93.81** ± 0.11	**93.47** ± 0.11
MobileNet V2	94.22 ± 0.15	**95.50** ± 0.09	**95.27** ± 0.10
Xception	90.51 ± 0.22	**93.25** ± 0.17	**92.59** ± 0.20
EffitientNet B0	95.10 ± 0.15	**96.23** ± 0.10	**96.11** ± 0.12

Table 4.6: Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR10 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

and WideResNet 28-10 models respectively.

We run experiments with Mixup augmentation method on CIFAR100 dataset with ShuffleNet V2 (2.0x), MobileNet V2, AlexNet, Xception, ResNet-50 models and results are reported in Table 4.13. The results are reported with the same experimental setup is reported in the CIFAR section. From table 4.13, it is clear that the proposed activations perform better than the baseline activations in all the models.

ImageNet-1k

We also evaluate the performance of proposed and baseline activation functions on bookmarking the ImageNet-1k dataset. The dataset consists of 1,281,167 training images and 50,000 validation images with 1000 classes. The images have a resolution of 224×224. Results are reported on Table 4.15 with ShuffleNet V2 (1.0x) and ResNet-50

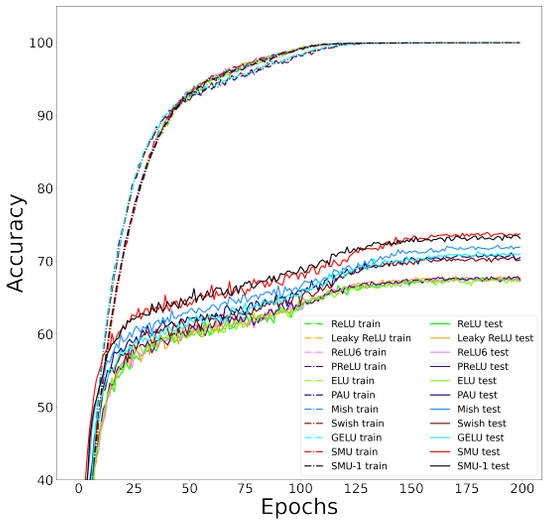

Figure 4.8: Top-1 train and test accuracy curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

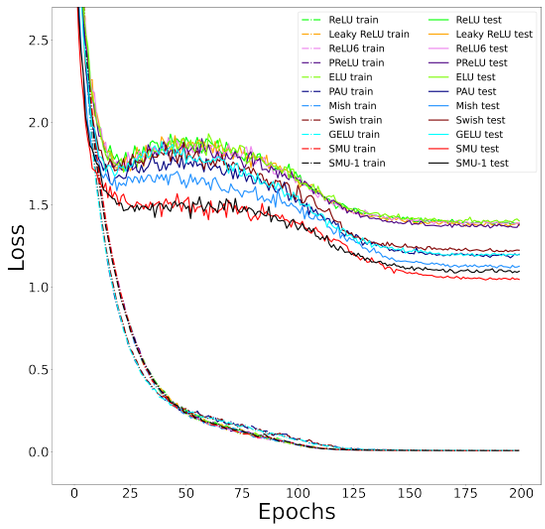

Figure 4.9: Top-1 train and test loss curves for SMU, SMU-1 and other baseline activation functions on CIFAR100 dataset with ShuffleNet V2 (2.0x) model.

Activation Function	Alex Net	Shuffle Net V1	Google Net	Inception V3	Dense Net 121	WideRes Net 28-10	Squeeze Net	VGG 16	LeNet
SMU	**61.27** ±0.21	**69.15** ±0.22	**74.61** ±0.25	**77.52** ±0.24	**78.57** ±0.23	**78.89** ±0.24	**68.51** ±0.24	**73.26** ±0.22	47.20 ±0.25
SMU-1	**60.98** ±0.23	**68.71** ±0.22	**74.29** ±0.24	**76.88** ±0.23	**78.01** ±0.24	**78.30** ±0.23	**68.07** ±0.25	**72.79** ±0.23	47.03 ±0.24
ReLU	54.89 ±0.28	65.79 ±0.29	72.52 ±0.30	74.12 ±0.27	75.81 ±0.28	76.45 ±0.26	66.22 ±0.29	71.87 ±0.30	45.54 ±0.28
Leaky ReLU	55.26 ±0.27	65.99 ±0.30	72.42 ±0.31	74.49 ±0.28	75.93 ±0.27	76.61 ±0.27	66.15 ±0.27	71.92 ±0.29	45.77 ±0.29
ReLU6	55.89 ±0.26	66.19 ±0.28	72.47 ±0.28	74.51 ±0.25	75.98 ±0.28	76.71 ±0.27	66.39 ±0.26	71.95 ±0.28	45.79 ±0.27
PReLU	55.47 ±0.29	65.87 ±0.32	72.69 ±0.29	74.39 ±0.30	76.06 ±0.29	76.71 ±0.27	66.35 ±0.28	71.96 ±0.32	45.59 ±0.31
ELU	55.91 ±0.26	65.72 ±0.28	72.92 ±0.28	74.65 ±0.26	75.72 ±0.25	76.25 ±0.26	66.39 ±0.28	71.79 ±0.30	46.02 ±0.28
Softplus	54.99 ±0.39	65.11 ±0.38	71.81 ±0.38	74.25 ±0.35	75.19 ±0.35	75.42 ±0.37	65.73 ±0.35	70.92 ±0.32	44.12 ±0.39
GELU	57.32 ±0.26	67.22 ±0.25	73.16 ±0.26	75.66 ±0.26	76.68 ±0.26	77.07 ±0.25	66.99 ±0.29	71.88 ±0.27	47.27 ±0.25
Swish	57.55 ±0.27	67.01 ±0.26	73.32 ±0.26	75.47 ±0.28	76.51 ±0.29	77.35 ±0.24	66.56 ±0.27	71.94 ±0.28	47.34 ±0.23
PAU	57.35 ±0.29	67.45 ±0.28	73.68 ±0.27	75.85 ±0.31	76.72 ±0.28	77.02 ±0.26	66.89 ±0.24	71.79 ±0.25	47.30 ±0.29
Mish	58.22 ±0.23	67.85 ±0.24	73.97 ±0.24	76.29 ±0.25	77.25 ±0.24	77.45 ±0.23	67.35 ±0.23	72.45 ±0.22	**47.42** ±0.27

Table 4.7: Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	SeNet 18	SeNet 34	SeNet 50	Res-Next	Xcep-tion	EffitientNet B0
Leaky ReLU	62.25 ±0.33	65.39 ±0.34	67.39 ±0.29	67.79 ±0.29	74.51 ±0.23	75.14 ±0.24	76.23 ±0.22	74.58 ±0.23	71.01 ±0.26	76.81 ±0.28
ReLU6	62.39 ±0.30	65.71 ±0.29	67.65 ±0.29	68.10 ±0.26	74.69 ±0.20	75.34 ±0.22	76.61 ±0.21	74.65 ±0.24	71.39 ±0.23	76.67 ±0.24
PReLU	62.01 ±0.34	65.10 ±0.33	67.37 ±0.34	67.99 ±0.29	74.39 ±0.27	75.19 ±0.23	76.39 ±0.24	74.48 ±0.26	71.19 ±0.28	76.68 ±0.30
ELU	62.61 ±0.30	65.60 ±0.28	67.71 ±0.27	67.91 ±0.31	74.79 ±0.22	75.10 ±0.21	76.49 ±0.23	74.69 ±0.22	71.45 ±0.24	76.71 ±0.30
Softplus	61.87 ±0.35	64.45 ±0.37	67.19 ±0.32	68.79 ±0.30	74.36 ±0.34	74.78 ±0.34	75.22 ±0.37	74.31 ±0.35	71.30 ±0.39	76.56 ±0.36
GELU	64.40 ±0.26	66.79 ±0.23	69.79 ±0.29	70.10 ±0.28	74.82 ±0.19	76.20 ±0.21	77.20 ±0.21	75.17 ±0.24	72.07 ±0.22	77.31 ±0.22
Swish	63.79 ±0.25	66.99 ±0.25	69.59 ±0.27	70.29 ±0.24	74.62 ±0.19	75.77 ±0.22	76.89 ±0.24	75.17 ±0.25	72.19 ±0.21	77.17 ±0.20
PAU	64.10 ±0.26	66.77 ±0.27	69.52 ±0.25	70.54 ±0.26	74.89 ±0.20	75.92 ±0.24	77.10 ±0.23	75.66 ±0.26	72.62 ±0.27	77.41 ±0.24
Mish	64.91 ±0.24	67.78 ±0.24	70.44 ±0.25	71.49 ±0.22	75.32 ±0.19	76.52 ±0.23	77.69 ±0.23	76.20 ±0.24	73.49 ±0.22	78.15 ±0.22

Table 4.8: This is an extension to the Table-4.5 (4.5). We report Top-1 test accuracy (in %) on CIFAR100 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. SF V2 stands for ShuffleNet v2.

Activation Function	ResNet 18	ResNet 34	ResNet 50	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50	MobileNet V1	MobileNet V2
Leaky ReLU	73.12 ±0.25	73.41 ±0.28	74.19 ±0.25	73.29 ±0.23	73.33 ±0.24	74.02 ±0.24	71.22 ±0.26	74.03 ±0.25
ReLU6	73.35 ±0.24	73.59 ±0.26	74.23 ±0.23	73.47 ±0.23	73.56 ±0.22	74.46 ±0.23	71.56 ±0.24	74.51 ±0.23
PReLU	73.02 ±0.27	73.52 ±0.29	74.32 ±0.28	73.21 ±0.25	73.45 ±0.26	74.29 ±0.25	71.41 ±0.29	74.45 ±0.30
ELU	73.42 ±0.24	73.68 ±0.26	74.48 ±0.24	73.32 ±0.20	73.49 ±0.25	74.44 ±0.25	71.32 ±0.24	74.22 ±0.23
Softplus	72.86 ±0.39	73.20 ±0.38	74.10 ±0.40	72.99 ±0.41	73.10 ±0.35	73.96 ±0.38	71.04 ±0.38	74.27 ±0.36
GELU	73.89 ±0.22	74.10 ±0.25	75.59 ±0.22	74.98 ±0.23	74.41 ±0.21	74.92 ±0.22	71.74 ±0.22	75.01 ±0.23
Swish	73.68 ±0.23	74.17 ±0.24	75.35 ±0.24	75.12 ±0.25	74.81 ±0.22	75.10 ±0.21	71.92 ±0.21	75.15 ±0.22
PAU	74.10 ±0.20	74.44 ±0.22	75.87 ±0.21	74.92 ±0.21	74.72 ±0.19	75.68 ±0.17	71.83 ±0.22	75.19 ±0.19
Mish	74.59 ±0.20	74.70 ±0.21	76.22 ±0.22	75.11 ±0.23	75.34 ±0.21	76.98 ±0.19	72.24 ±0.20	75.45 ±0.20

Table 4.9: This is an extension to the Table-4.5 (4.5). We report Top-1 test accuracy (in %) on CIFAR100 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	ResNet 18	ResNet 34	ResNet 50	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50	MobileNet V1	MobileNet V2
Leaky ReLU	94.00 ±0.25	94.18 ±0.24	94.29 ±0.24	93.51 ±0.20	94.29 ±0.22	94.32 ±0.22	92.54 ±0.21	94.10 ±0.19
ReLU6	94.19 ±0.26	94.20 ±0.25	94.26 ±0.27	93.69 ±0.21	94.19 ±0.25	94.52 ±0.23	92.69 ±0.20	94.21 ±0.20
PReLU	94.22 ±0.28	94.29 ±0.29	94.17 ±0.27	93.58 ±0.23	94.31 ±0.25	94.48 ±0.28	92.50 ±0.20	94.29 ±0.23
ELU	94.15 ±0.23	94.24 ±0.22	94.20 ±0.24	93.59 ±0.22	94.42 ±0.24	94.45 ±0.20	92.69 ±0.21	94.04 ±0.19
Softplus	93.82 ±0.29	93.99 ±0.31	93.77 ±0.31	93.09 ±0.28	94.01 ±0.35	94.08 ±0.32	92.01 ±0.32	93.91 ±0.27
GELU	94.38 ±0.22	94.41 ±0.23	94.59 ±0.23	93.70 ±0.21	94.24 ±0.25	94.69 ±0.23	92.81 ±0.20	94.20 ±0.16
Swish	94.31 ±0.21	94.32 ±0.20	94.64 ±0.22	93.80 ±0.21	94.14 ±0.24	94.61 ±0.23	92.69 ±0.22	94.22 ±0.17
PAU	94.40 ±0.20	94.46 ±0.22	94.59 ±0.22	93.84 ±0.20	94.29 ±0.22	94.73 ±0.24	93.01 ±0.15	94.54 ±0.13
Mish	94.52 ±0.23	94.39 ±0.22	94.79 ±0.22	93.78 ±0.22	94.51 ±0.22	94.81 ±0.24	92.78 ±0.20	94.77 ±0.18

Table 4.10: This is an extension to the Table-4.6 (4.6). We report Top-1 test accuracy (in %) on CIFAR10 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	Alex Net	Shuffle Net V1	Google Net	Inception V3	Dense Net 121	WideRes Net 28-10	Squeeze Net	VGG 16	LeNet
SMU	**87.25** ±0.15	**92.42** ±0.14	**94.10** ±0.17	**95.59** ±0.14	**96.07** ±0.12	**96.23** ±0.14	**91.77** ±0.16	**94.54** ±0.14	77.66 ±0.16
SMU-1	**86.77** ±0.16	**92.01** ±0.15	**93.69** ±0.16	**95.11** ±0.15	**95.65** ±0.12	**95.71** ±0.13	**91.38** ±0.15	**94.32** ±0.15	77.39 ±0.16
ReLU	84.10 ±0.20	91.34 ±0.19	92.91 ±0.18	94.04 ±0.18	94.77 ±0.19	95.08 ±0.21	90.59 ±0.20	93.59 ±0.18	75.80 ±0.21
Leaky ReLU	84.22 ±0.22	91.56 ±0.20	92.79 ±0.17	94.29 ±0.22	94.68 ±0.22	95.01 ±0.20	90.71 ±0.20	93.71 ±0.19	75.99 ±0.20
ReLU6	84.79 ±0.19	91.68 ±0.18	92.97 ±0.16	94.21 ±0.19	94.59 ±0.20	95.39 ±0.20	90.87 ±0.19	93.70 ±0.17	75.88 ±0.18
PReLU	84.30 ±0.24	91.74 ±0.23	92.91 ±0.24	94.45 ±0.20	94.59 ±0.23	95.10 ±0.20	90.79 ±0.23	93.58 ±0.22	75.90 ±0.21
ELU	84.89 ±0.19	91.89 ±0.18	92.99 ±0.16	94.45 ±0.17	94.72 ±0.18	95.23 ±0.17	90.87 ±0.15	93.78 ±0.16	75.88 ±0.18
Softplus	84.01 ±0.30	91.10 ±0.29	92.56 ±0.32	94.17 ±0.31	94.54 ±0.29	94.89 ±0.28	90.55 ±0.33	93.39 ±0.29	75.45 ±0.35
GELU	85.02 ±0.19	91.77 ±0.18	93.36 ±0.18	94.32 ±0.17	94.71 ±0.20	95.19 ±0.18	90.89 ±0.16	93.64 ±0.16	77.71 ±0.19
Swish	85.19 ±0.18	91.49 ±0.20	93.26 ±0.19	94.40 ±0.19	94.69 ±0.17	95.47 ±0.17	91.12 ±0.19	93.68 ±0.17	77.70 ±0.18
PAU	84.91 ±0.20	91.95 ±0.21	93.20 ±0.19	94.32 ±0.23	94.50 ±0.22	95.07 ±0.20	90.51 ±0.19	93.50 ±0.21	77.68 ±0.20
Mish	85.78 ±0.17	91.96 ±0.15	93.29 ±0.17	94.49 ±0.16	95.03 ±0.13	95.39 ±0.16	91.14 ±0.16	93.77 ±0.17	**77.79** ±0.15

Table 4.11: Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR10 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	SeNet 18	SeNet 34	SeNet 50	Res-Next	Xcep-tion	EffitientNet B0
Leaky ReLU	88.32 ±0.24	91.20 ±0.26	91.24 ±0.24	91.70 ±0.24	94.18 ±0.24	94.52 ±0.23	94.51 ±0.21	93.25 ±0.20	90.81 ±0.25	95.35 ±0.15
ReLU6	88.52 ±0.22	91.15 ±0.23	91.32 ±0.20	91.64 ±0.22	94.39 ±0.22	94.50 ±0.24	94.61 ±0.22	93.49 ±0.21	91.20 ±0.22	95.40 ±0.16
PReLU	88.28 ±0.24	91.02 ±0.22	91.29 ±0.24	91.77 ±0.24	94.35 ±0.24	94.57 ±0.25	94.62 ±0.24	93.35 ±0.24	91.07 ±0.23	95.37 ±0.19
ELU	88.20 ±0.22	91.17 ±0.26	91.40 ±0.22	91.81 ±0.24	94.22 ±0.22	94.42 ±0.24	94.71 ±0.22	93.52 ±0.23	91.45 ±0.22	95.19 ±0.19
Softplus	87.95 ±0.30	90.42 ±0.30	91.01 ±0.28	91.00 ±0.30	93.82 ±0.29	94.05 ±0.30	94.22 ±0.27	93.10 ±0.29	90.56 ±0.27	95.07 ±0.25
GELU	88.92 ±0.20	91.62 ±0.24	91.77 ±0.20	92.29 ±0.19	94.49 ±0.20	94.77 ±0.20	94.79 ±0.16	93.61 ±0.20	91.99 ±0.22	95.45 ±0.15
Swish	89.04 ±0.20	91.71 ±0.22	91.81 ±0.20	92.20 ±0.19	94.30 ±0.18	94.69 ±0.18	94.55 ±0.17	93.61 ±0.19	91.69 ±0.19	95.56 ±0.16
PAU	89.18 ±0.21	91.70 ±0.24	92.20 ±0.20	92.31 ±0.19	94.32 ±0.21	94.77 ±0.22	94.70 ±0.20	93.50 ±0.19	91.91 ±0.22	95.49 ±0.15
Mish	89.42 ±0.20	91.98 ±0.18	92.18 ±0.17	92.47 ±0.18	94.49 ±0.19	94.81 ±0.18	94.97 ±0.15	93.89 ±0.17	92.07 ±0.20	95.70 ±0.12

Table 4.12: This is an extension to the Table-4.6 (4.5). We report Top-1 test accuracy (in %) on CIFAR10 dataset for baseline functions for the mean of 15 different runs. mean±std is reported in the table. SF V2 stands for ShuffleNet v2.

Activation Function	Shuffle Net V2 (2.0x)	ResNet 50	Xception	Alex Net	Mobile Net V2
ReLU	70.21 ± 0.23	75.61 ± 0.26	72.10 ± 0.20	55.80 ± 0.27	75.72 ± 0.23
Leaky ReLU	70.09 ± 0.25	75.74 ± 0.27	72.22 ± 0.22	56.10 ± 0.28	75.81 ± 0.25
PReLU	70.17 ± 0.24	75.82 ± 0.28	72.18 ± 0.24	56.52 ± 0.26	75.98 ± 0.27
ReLU6	70.21 ± 0.23	76.14 ± 0.25	72.35 ± 0.19	56.69 ± 0.26	75.87 ± 0.22
ELU	70.34 ± 0.24	76.15 ± 0.26	72.41 ± 0.23	56.97 ± 0.25	75.79 ± 0.25
Softplus	69.91 ± 0.26	75.51 ± 0.30	71.94 ± 0.26	55.65 ± 0.35	75.60 ± 0.27
Swish	73.64 ± 0.21	76.80 ± 0.24	73.45 ± 0.20	58.77 ± 0.24	76.67 ± 0.21
Mish	74.25 ± 0.22	77.30 ± 0.24	74.34 ± 0.21	59.87 ± 0.25	77.02 ± 0.22
GELU	73.51 ± 0.21	76.85 ± 0.25	73.71 ± 0.18	58.50 ± 0.26	76.61 ± 0.22
PAU	73.85 ± 0.20	77.07 ± 0.24	73.87 ± 0.20	58.80 ± 0.25	76.81 ± 0.21
SMU	**75.78** ± 0.20	**78.71** ± 0.24	**75.30** ± 0.18	**62.42** ± 0.23	**77.83** ± 0.20
SMU-1	**75.01** ± 0.21	**77.81** ± 0.24	**74.84** ± 0.20	**61.93** ± 0.25	**77.49** ± 0.22

Table 4.13: Comparison between SMU, SMU-1 activations and other baseline activations on CIFAR100 dataset for image classification problem with Mixup augmentation method. We report Top-1 test accuracy (in %) for the mean of 15 different runs. mean±std is reported in the table.

Activation Function	DenseNet-121	ResNet-18	ResNet-50	WideResNet 28-10
ReLU	63.31 ± 0.47	59.12 ± 0.44	61.23 ± 0.46	63.74 ± 0.40
Leaky ReLU	63.63 ± 0.48	59.40 ± 0.44	61.29 ± 0.44	63.61 ± 0.42
PReLU	63.71 ± 0.46	59.59 ± 0.42	61.35 ± 0.44	63.78 ± 0.44
ReLU6	63.54 ± 0.49	59.49 ± 0.46	61.41 ± 0.44	63.72 ± 0.43
ELU	63.51 ± 0.46	59.34 ± 0.44	61.49 ± 0.43	63.72 ± 0.43
Softplus	63.01 ± 0.57	59.01 ± 0.57	60.93 ± 0.57	63.01 ± 0.59
Swish	64.21 ± 0.40	60.05 ± 0.40	61.79 ± 0.41	64.58 ± 0.41
Mish	64.47 ± 0.40	60.21 ± 0.39	62.07 ± 0.42	64.79 ± 0.38
GELU	64.34 ± 0.42	60.21 ± 0.41	61.66 ± 0.42	64.39 ± 0.40
PAU	64.04 ± 0.43	60.37 ± 0.39	61.72 ± 0.41	64.42 ± 0.40
SMU	**65.87** ± 0.37	**61.35** ± 0.35	**63.54** ± 0.40	**66.52** ± 0.35
SMU-1	**65.09** ± 0.38	**60.93** ± 0.38	**62.79** ± 0.40	**65.25** ± 0.37

Table 4.14: Comparison between SMU, SMU-1 activations and other baseline activations on Tiny ImageNet dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 10 different runs. mean±std is reported in the table.

models. We use four NVIDIA V100 GPUs with 32GB RAM each to run these experiments. We trained the models up to 600k iterations with a batch size of 256 and SGD optimizer, 0.9 momentum, $5e^{-4}$ weight decay rate.

Activation Function	ShuffleNet V2 (1.0x)	ResNet-50
ReLU	69.21	75.52
Leaky ReLU	69.28	75.67
PReLU	69.01	75.40
ReLU6	69.45	75.70
ELU	69.49	75.62
Softplus	69.01	75.30
Swish	70.35	76.06
Mish	70.53	76.45
GELU	70.12	76.01
PAU	70.28	76.14
SMU	**71.93**	**77.48**
SMU-1	**71.17**	**76.89**

Table 4.15: Top-1 accuracy reported on ImageNet-1k dataset.

4.5.2 Object Detection

In this section, we report results on object detection problem on Pascal VOC dataset (Everingham *et al.* (2010)) with Single Shot MultiBox Detector(SSD) 300 model (Liu *et al.* (2016)) and we consider VGG-16 (with batch-normalization) (Simonyan and Zisserman (2015)) as the backbone network. We use VOC2007 & VOC2012 as train data and VOC2007 as the test dataset. The dataset contains 20 different objects. We consider a batch size of 8, 0.001 initial learning rate and decay the learning rate as reported in (Liu *et al.* (2016)). We use SGD (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained networks upto 120000 iterations. We do not consider any pre-trained weight. We report the mean average precision (mAP) in Table 4.16 for the mean of 10 different runs. Replacing ReLU by SMU, we got a 1% improvement in mAP in the test dataset.

Activation Function	mAP
ReLU	77.2 ± 0.14
Leaky ReLU	77.2 ± 0.13
PReLU	77.2 ± 0.16
ReLU6	77.1 ± 0.15
ELU	75.1 ± 0.18
Softplus	74.2 ± 0.25
Swish	77.5 ± 0.11
Mish	77.6 ± 0.11
GELU	77.5 ± 0.12
PAU	77.4 ± 0.14
SMU	**78.2** ± 0.09
SMU-1	**77.8** ± 0.11

Table 4.16: Comparison between SMU, SMU-1 activations and other baseline activations on Pascal VOC dataset for object detection problem. We report mAP for the mean of 10 different runs. mean$\pm$std is reported in the table.

4.5.3 Semantic Segmentation

In this section, we report experimental results on semantic segmentation problems on the popular CityScapes dataset (Cordts *et al.* (2016)). CityScapes (Cordts *et al.* (2016)) is a popular dataset consisting of diverse urban street scenes across 50 different cities at varying times of the year, as well as ground truths for semantic segmentation, instance-level segmentation. Label annotations for segmentation tasks span across 30+ classes. We consider U-net model (Ronneberger *et al.* (2015)) as the segmentation framework. The model is trained with adam optimizer (Kingma and Ba (2015)), $5e^{-3}$ learning rate, a batch size 32 up to 250 epochs. We report the mean of 10 different runs for Pixel Accuracy and the mean Intersection-Over-Union (mIOU) on test data on table 4.17.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.49 $\pm$ 0.46	69.31 $\pm$ 0.28
Leaky ReLU	79.41 $\pm$ 0.41	69.64 $\pm$ 0.42
PReLU	78.95 $\pm$ 0.42	68.88 $\pm$ 0.41
ReLU6	79.58 $\pm$ 0.41	69.70 $\pm$ 0.42
ELU	79.48 $\pm$ 0.50	68.19 $\pm$ 0.40
Softplus	78.45 $\pm$ 0.52	68.08 $\pm$ 0.49
Swish	80.22 $\pm$ 0.46	69.81 $\pm$ 0.30
Mish	80.59 $\pm$ 0.44	70.12 $\pm$ 0.30
GELU	80.14 $\pm$ 0.37	69.59 $\pm$ 0.40
PAU	79.89 $\pm$ 0.39	69.31 $\pm$ 0.44
SMU	**81.79** $\pm$ 0.36	**71.11** $\pm$ 0.30
SMU-1	**80.75** $\pm$ 0.41	**70.55** $\pm$ 0.30

Table 4.17: Comparison between SMU, SMU-1 activations and other baseline activations on CityScapes dataset for semantic segmentation problem. We report pixel accuracy and mIOU for the mean of 10 different runs. mean$\pm$std is reported in the table.

4.5.4 Machine Translation

In this section, we report the result on the machine translation problem. This problem deals with the translation of text or speech data from one language to another language without the help of any human being. The WMT 2014 English$\rightarrow$German dataset is used for our experiment. The database contains 4.5 million training sentences. We use an attention-based (Vaswani *et al.* (2017)) 8-head transformer network with Adam optimizer (Kingma and Ba (2015)), 0.1 dropout rate (Srivastava *et al.* (2014)), and

train up to 100000 steps. Other hyperparameters are kept similar as mentioned in the original paper (Vaswani *et al.* (2017)). We evaluate the network performance on the newstest2014 dataset using the BLEU score metric. We report the mean of 10 different runs on Table 4.18 on the test dataset(newstest2014). The table shows that the results are stable on different runs (mean±std), and we got around 0.6% boost in BLEU score for SMU compared to ReLU.

Activation Function	BLEU Score
ReLU	26.2 ± 0.14
Leaky ReLU	26.3 ± 0.15
PReLU	26.2 ± 0.18
ReLU6	26.1 ± 0.14
ELU	25.1 ± 0.14
Softplus	23.6 ± 0.18
Swish	26.4 ± 0.11
Mish	26.3 ± 0.12
GELU	26.4 ± 0.15
PAU	26.3 ± 0.15
SMU	$\mathbf{26.8} \pm 0.11$
SMU-1	$\mathbf{26.6} \pm 0.10$

Table 4.18: Comparison between SMU, SMU-1 activations and other baseline activations on WMT2014 dataset for machine translation problem. We report BLEU score for the mean of 10 different runs. mean±std is reported in the table.

4.6 Baseline Table

SMU and SMU-1 are novel activation functions constructed using the smoothing of maximum function. For a detailed comparison, we report a summary of all the experiments in Table 4.19 given in earlier sections. It is pretty clear from Table 4.19 that the proposed functions outperform baseline functions almost in all experiments.

4.7 Computational Time Comparison

In this section, we report the computational Time Comparison for SMU, SMU-1, and baseline activation functions. We report results in Table 4.20 for the mean of 100 runs on a 32×32 RGB image in ResNet-18 He *et al.* (2015*a*) model for both forward and

Baselines	ReLU	Leaky ReLU	ELU	Softplus	PReLU	ReLU6	Swish	Mish	GELU	PAU
SMU $>$ Baseline	80	80	80	80	80	80	77	76	77	78
SMU $=$ Baseline	0	0	0	0	0	0	0	0	0	0
SMU $<$ Baseline	0	0	0	0	0	0	3	4	3	2
SMU-1 $>$ Baseline	80	80	80	80	80	80	77	76	77	78
SMU-1 $=$ Baseline	0	0	0	0	0	0	0	0	0	0
SMU-1 $<$ Baseline	0	0	0	0	0	0	3	4	3	2

Table 4.19: Baseline table for SMU. These numbers represent the total number of models in which SMU underperform, equal or outperform compared to the baseline activation functions

backward pass. The experiments are conducted on an NVIDIA Tesla V100 GPU with 32GB RAM. It is noticeable from the experiment section and Table 4.20 that there is a small trade-off between the computational time and model performances compared to ReLU or its variants. The proposed activations have significantly boosted the model performance though it has slightly higher computational time (due to non-linearity and the trainable parameter μ) than ReLU or its variants. In contrast, the computational time is similar to popular non-linear activations like Swish, Mish & GELU and much better than PAU, while model performance at the same time is comparatively much better than these four popular non-linear activations in almost all cases.

Activation Function	Forward Pass	Backward Pass
ReLU	6.43 ± 0.31 μs	6.28 ± 0.74 μs
Leaky ReLU	6.49 ± 0.41 μs	6.41 ± 0.95 μs
PReLU	$8.20 \pm 1.57\mu$s	9.26 ± 1.86 μs
ReLU6	6.45 ± 0.45 μs	6.41 ± 0.91 μs
ELU	6.51 ± 0.50 μs	6.42 ± 0.88 μs
Softplus	6.49 ± 0.49 μs	6.40 ± 0.55 μs
Mish	10.02 ± 1.79 μs	11.97 ± 1.75 μs
GELU	10.75 ± 1.49 μs	12.49 ± 1.77 μs
Swish	10.47 ± 1.10 μs	12.61 ± 1.22 μs
PAU	18.45 ± 3.40 μs	25.99 ± 5.06 μs
SMU	10.74 ± 1.29 μs	12.95 ± 1.54 μs
SMU-1	9.68 ± 1.81 μs	11.98 ± 1.49 μs

Table 4.20: Runtime comparison for the forward and backward passes for SMU and SMU-1 and other baseline activation functions for a $32\times$ 32 RGB image in ResNet-18 model.

4.8 Conclusion

This work uses the maximum smoothing technique to approximate Leaky ReLU, a well-established activation function (not differentiable at 0) by two smooth functions. These two functions are named SMU and SMU-1 and are being proposed as potential candidates for activation functions. Our experimental evaluation shows that the proposed functions beat the traditional activation functions in well-known deep learning problems and have the potential to replace them. An extensive amount of experiments are being conducted in different datasets on four different deep learning problems to show the efficacy of the proposed activation functions. From the running time table, it is clear that the proposed activation functions have similar ruining time like Swish, GELU, and Mish, while both SMU and SMU-1 improves network performance compared to these three non-linear activation functions.

CHAPTER 5

ErfAct and Pserf [1]

5.1 Introduction

This chapter, in particular, deals with the hand-designed activation function. ReLU and Leaky ReLU are popular hand-designed activation functions. Though both are non-differentiable at the origin, differentiability is an important property in the deep neural network. Swish is a smooth activation function and approximation of the ReLU activation function. In this chapter, two new smooth activations have been proposed, which are approximations by a smooth function of the ReLU activation function.

The choice of activation function in a deep learning architecture can have a significant impact on the training and performance of the neural network. The machine learning community has so far relied on hand-designed activations like ReLU Nair and Hinton (2010), Leaky ReLU (Maas *et al.* (2013*a*)) or their variants. ReLU, in particular, remains widely popular due to faster training times and decent performance. However, evidence suggests that considerable gains can be made when more sophisticated activation functions are used to design networks. For example, activation functions such as ELU (Clevert *et al.* (2016)), Parametric ReLU (PReLU) (He *et al.* (2015*b*)), ReLU6 (Krizhevsky (2010)), PAU (Molina *et al.* (2020)), ACON (Ma *et al.* (2021)), Mish (Misra (2020)), GELU (Hendrycks and Gimpel (2020)), Swish (Ramachandran *et al.* (2017)), Serf (Nag and Bhattacharyya (2021)) etc. have appeared as powerful contenders to the traditional ones. ReLU remains a go-to choice in research and practice. However, it has certain well-documented shortcomings, such as non-zero mean, non-differentiability and negative missing, which leads to the infamous vanishing gradients problem (also known as the dying ReLU problem). Worth noting that prior to the introduction of ReLU, Tanh and Sigmoid were popularly used, but performance gains and training time gains achieved by ReLU led to their decline.

[1] This chapter is a slightly modified version of the paper accepted at AAAI conference Biswas *et al.* (2021*c*).

5.2 Related Works and Motivation

The newer activation functions are obtained by combining well-known functions with simple forms in various ways, often using hyper-parameters or trainable parameters. In the case of trainable parameters, we optimize them during the training process itself, yielding networks that are better fitted. In the case of trainable parameters, note that the actual activation function curve may change in different layers during backpropagation. For example, SiLU (Elfwing *et al.* (2017)) shows good performance over known activation functions. In contrast, Swish (Ramachandran *et al.* (2017)) is a trainable version of SiLU, which is a non-linear, non-monotonic, smooth activation function. Swish, PReLU, PAU, and ACON are trainable activation functions, among the other activation functions. Swish is a non-monotonic activation function and shows promise across a variety of deep learning tasks. Mish is one of the popular functions proposed recently and gained popularity due to its effectiveness in object detection tasks on the COCO dataset (Lin *et al.* (2015)) in Yolo (Bochkovskiy *et al.* (2020)) models. GELU is very similar to Swish and gained attention due to its effectiveness in computer vision and natural language processing tasks. It is also used in popular architectures like GPT-2 (Radford *et al.* (2019)) and GPT-3 (Brown *et al.* (2020)). Apart from using a combination of known functions, a somewhat fundamentally different technique to construct activation functions is to use perturbation or approximations to well-known activation functions to remove some shortcomings yet retain the positive aspects. Recent successful examples where this strategy was employed include PAU, which is activation based on an approximation of Leaky ReLU by rational polynomials were constructed.

5.3 Research contribution

Motivated by these works, we have proposed two activation functions with trainable parameters; we call them ErfAct and Pserf and have shown that they are more effective than conventional activation functions like ReLU, Leaky ReLU, PReLU, ReLU6, Swish, Mish or GELU in a wide range of standard deep learning problems. We summarize the chapter as follows:

- We have proposed two new novel trainable activation functions, which are approximation by a smooth function of ReLU.

- In a wide range of deep learning tasks, the proposed functions outperform widely used activation functions.

5.4 ErfAct and Pserf

We present, ErfAct and Parametric-Serf (Pserf), two novel trainable activation functions which outperforms the widely used activations and has the potential to replace them. ErfAct and Pserf is defined as

$$\text{ErfAct} : \mathcal{F}_1(x;\alpha,\beta) := x \ \text{erf}(\alpha e^{\beta x}), \tag{5.1}$$

$$\text{Pserf} : \ \mathcal{F}_2(x;\gamma,\delta) := x \ \text{erf}(\gamma ln(1+e^{\delta x})) \tag{5.2}$$

where α, β, γ, and δ are trainable parameters (they can be used as hyper-parameters as well) and 'erf' is the error function also known as the Gauss error function and defined as

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} \, dt \tag{5.3}$$

The corresponding derivatives of the proposed activations are

$$\frac{d}{dx}\mathcal{F}_1(x;\alpha,\beta) = \text{erf}(\alpha e^{\beta x}) + \frac{2x\alpha\beta}{\sqrt{\pi}} e^{\beta x} e^{-(\alpha e^{\beta x})^2} \tag{5.4}$$

$$\begin{aligned}\frac{d}{dx}\mathcal{F}_2(x;\gamma,\delta) =& \text{erf}(\gamma ln(1+e^{\delta x})) \\ &+ \frac{2x\gamma\delta}{\sqrt{\pi}} \frac{e^{\delta x}}{1+e^{\delta x}} e^{-(\gamma ln(1+e^{\delta x}))^2}\end{aligned} \tag{5.5}$$

where

$$\frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}} e^{-x^2} \tag{5.6}$$

ErfAct and Pserf are non-monotonic, zero-centered, continuously differentiable, unbounded above but bounded below, and trainable functions. Figures 5.1 and 5.2 show

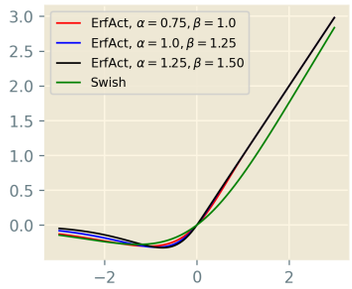

Figure 5.1: Swish and ErfAct activation for different values of α and β

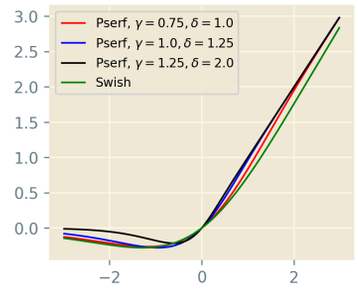

Figure 5.2: Swish and Pserf activation for different values of γ and δ

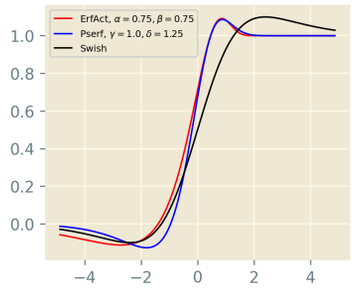

Figure 5.3: First order derivative of ErfAct, Pserf, and Swish

the plots for $\mathcal{F}_1(x;\alpha,\beta)$ and $\mathcal{F}_2(x;\gamma,\delta)$ activation functions for different values of α, β, and γ,δ respectively. A comparison between the first derivative of $\mathcal{F}_1(x;\alpha,\beta)$, $\mathcal{F}_2(x;\gamma,\delta)$, and Swish are given in Figures 5.3, different values of α, β, and γ,δ respectively. From the figures 5.1 and 5.2 it is evident that the parameters α, β, and γ,δ controls the slope of the curves for the proposed activations in both positive and negative axis. The proposed functions converges to some known functions for specific values of the parameters. For example, $\mathcal{F}_1(x;0,\beta)$, $\mathcal{F}_2(x;0,\delta)$ are zero function while $\mathcal{F}_1(x;\alpha,0)$, $\mathcal{F}_2(x;\gamma,0)$ are linear functions. In particular, $\mathcal{F}_2(x;1,1)$ share the equivalent form as Serf (Nag and Bhattacharyya (2021)) which is a non-parametric form of Pserf. Also, The proposed functions can be seen as approximation by a smooth function of ReLU.

$$\lim_{\beta\to\infty} \mathcal{F}_1(x;\alpha,\beta) = \mathrm{ReLU}(x), \quad \forall x \in \mathbb{R} \text{ for any fixed } \alpha > 0.$$

$$\lim_{\delta\to\infty} \mathcal{F}_2(x;\gamma,\delta) = \mathrm{ReLU}(x), \quad \forall x \in \mathbb{R} \text{ for any fixed } \gamma > 0.$$

For any K, a compact (closed and bounded) subset of $\mathbb{R}^n$, the set of neural networks with ErfAct (or Pserf) activation functions is dense in $C(K)$, the space of all continuous functions over K (see Molina *et al.* (2020)). This follows from the next proposition, as the proposed activation functions are not polynomials.

Proposition (Theorem 1.1 in Kidger and Lyons, 2019 Kidger and Lyons (2020)) :- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

5.5 Experiments

We have compared our proposed activations against ten popular standard activation functions on different datasets and models on standard deep learning problems like image classification, object detection, semantic segmentation, and machine translation. The experimental results show that ErfAct and Pserf outperform in most networks compared to the standard activations. For all our experiments, we have first initialized the parameters α, β for ErfAct and γ, δ for Pserf and then updated via the backpropagation (LeCun *et al.* (1989)) algorithm (see He *et al.* (2015*b*)) according to (5.7) and for a single layer, the gradient of a parameter ρ is:

$$\frac{\partial L}{\partial \rho} = \sum_x \frac{\partial L}{\partial f(x)} \frac{\partial f(x)}{\partial \rho} \tag{5.7}$$

where L is the objective function, $\rho \in \{\alpha, \beta, \gamma, \delta\}$ and $f(x) \in \{\mathcal{F}_1(x; \alpha, \beta), \mathcal{F}_2(x; \gamma, \delta)$. For all of our experiments, to make a fair comparison between all the activations, we have first trained a network with hyper-parameter settings with the ReLU activation function and then only replaced ReLU with proposed activation functions and other baseline activations.

5.5.1 Image Classification

We present a detailed experimental comparison on MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), SVHN (Netzer *et al.* (2011)), CIFAR10 (Krizhevsky (2009)), CIFAR100 (Krizhevsky (2009)), Tiny ImageNet (Le and Yang (2015)), and ImageNet-1k (Deng *et al.* (2009)) dataset for image classification problem. We have trained the datasets with different standard models and report the Top-1 accuracy. We have initialized the parameters $\alpha = 0.75,\ \beta = 0.75$ for ErfAct, and $\gamma = 1.25,\ \delta = 0.85$

for Pserf and update them according to (5.7).

MNIST, Fashion MNIST, and The Street View House Numbers (SVHN) Database:

We first evaluate our proposed activation functions on the MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), and SVHN (Netzer *et al.* (2011)) datasets with AlexNet (Krizhevsky *et al.* (2012)) and VGG-16 (Simonyan and Zisserman (2015)) (with batch-normalization) models and results for 10-fold mean accuracy are reported in Table 5.1 and Table 5.2 respectively. More detailed experiments on these datasets on LeNet (Lecun *et al.* (1998)) and a custom-designed CNN architecture and the results are reported on Table 5.3 & 5.4. The custom network is constructed with an 8-layer homogeneous custom convolutional neural network (CNN) architecture with 3×3 kernels and max-pooling layers with 2×2 kernels. We have used Channel depths of size 128 (twice), 64 (thrice), 32 (twice), with a dense layer of size 128, Max-pooling layer(thrice), and dropout (Srivastava *et al.* (2014)). We have applied batch-normalization (Ioffe and Szegedy (2015)) before the activation function layer. We don't use any data augmentation for MNIST or Fashion MNIST, while we use standard data augmentation like rotation, zoom, height shift, shearing for the SVHN dataset. From Table 5.1, 5.2, 5.3, and Table 5.4, it is clear that the proposed functions outperformed all the baseline activation functions in all the three datasets and the performance are stable clear from mean$\pm$standard deviation.

CIFAR:

Next we have considered more challenging datasets like CIFAR100 and CIFAR10 to compare the performance of baseline activations and ErfAct and Pserf. We have reported the Top-1 accuracy for both the datasets for mean of 12 different runs on Table 5.5 and Table 5.6 with VGG-16 (with batch-normalization) (Simonyan and Zisserman (2015)), PreActResNet-34 (PA-ResNet-34) (He *et al.* (2016)), Densenet-121 (DN-121) (Huang *et al.* (2016*a*)), MobileNet V2 (MN V2) (Sandler *et al.* (2019)), Resnet-50 (He *et al.* (2015*a*)), Inception V3 (IN-V3) (Szegedy *et al.* (2015*a*)), WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), and Shufflenet V2 (SF-V2 2.0x) (Ma *et al.* (2018)) models. A more detailed experiments on CIFAR10 and CIFAR100 datasets with EfficientNet B0 (EN-B0) (Tan and Le (2020)), LeNet (LN) (Lecun *et al.*

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.09 ± 0.10	93.22 ± 0.21	95.50 ± 0.22
Swish	99.30 ± 0.12	93.29 ± 0.22	95.59 ± 0.20
Leaky ReLU	99.15 ± 0.13	93.30 ± 0.22	95.50 ± 0.28
ELU	99.29 ± 0.13	93.20 ± 0.25	95.60 ± 0.20
Softplus	99.10 ± 0.14	93.18 ± 0.32	95.20 ± 0.37
Mish	99.27 ± 0.14	93.45 ± 0.32	95.60 ± 0.31
GELU	99.22 ± 0.12	93.40 ± 0.25	95.55 ± 0.27
PAU	99.31 ±0.10	93.47 ± 0.23	95.67 ± 0.26
PReLU	99.15 ± 0.16	93.37 ± 0.31	95.42 ± 0.39
ReLU6	99.11 ± 0.10	93.26 ± 0.26	95.47 ± 0.24
ErfAct	**99.51** ± 0.10	**93.79** ± 0.19	**95.87** ± 0.20
Pserf	**99.49** ± 0.10	**93.82** ± 0.19	**95.74** ± 0.22

Table 5.1: Comparison between different baseline activations and ErfAct and Pserf activations on MNIST, Fashion MNIST, and SVHN datasets in AlexNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.05 ± 0.11	93.13 ± 0.23	95.09 ± 0.26
Swish	99.09 ± 0.09	93.34 ± 0.21	95.29 ± 0.20
Leaky ReLU	99.02 ± 0.14	93.17 ± 0.28	95.24 ± 0.23
ELU	99.01 ± 0.15	93.12 ± 0.30	95.15 ± 0.28
Softplus	98.97 ± 0.14	92.98 ± 0.34	94.94 ± 0.30
Mish	99.18 ± 0.07	93.47 ± 0.27	95.12 ± 0.25
GELU	99.10 ± 0.09	93.41 ± 0.29	95.11 ± 0.24
PAU	99.07 ± 0.09	93.52 ± 0.24	95.23 ± 0.20
PReLU	99.01 ± 0.09	93.12 ± 0.27	95.14 ± 0.24
ReLU6	99.20 ± 0.08	93.25 ± 0.27	95.22 ± 0.20
ErfAct	**99.37** ± 0.06	**93.81** ± 0.20	**95.67** ± 0.18
Pserf	**99.38** ± 0.09	**93.87** ± 0.22	**95.66** ± 0.20

Table 5.2: Comparison between different baseline activations, ErfAct, and Pserf activations on MNIST, Fashion MNIST, and SVHN datasets on VGG-16 network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.09±0.10	92.92±0.21	95.10±0.22
Swish	99.20±0.09	93.04±0.23	95.21±0.23
Leaky ReLU(α = 0.01)	99.14±0.09	92.99±0.22	95.30±0.25
ELU	99.10±0.13	92.91±0.30	95.17±0.27
Softplus	98.95 ±0.17	92.72±0.28	95.08±0.37
Mish	99.32±0.10	93.12±0.21	95.33±0.21
GELU	99.28±0.09	93.19±0.22	95.22±0.24
PReLU	99.08±0.17	92.89±0.35	95.15±0.30
ReLU6	99.17±0.12	92.99±0.20	95.17±0.22
PAU	99.24±0.10	93.24±0.20	95.15±0.23
ErfAct	**99.42**±0.08	**93.42**±0.23	**95.49**±0.24
Pserf	**99.40**±0.08	**93.35**±0.20	**95.55**±0.23

Table 5.3: Comparison between different baseline activations and ErfAct and Pserf on MNIST, Fashion MNIST, and SVHN datasets with Custom designed network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	98.95±0.11	91.00±0.20	93.17±0.24
Swish	99.04±0.11	91.15±0.22	93.22±0.21
Leaky ReLU(α = 0.01)	99.02±0.10	91.05±0.20	93.25±0.24
ELU	98.95±0.12	91.98±0.28	93.11±0.24
Softplus	98.81 ±0.14	90.81±0.29	93.08±0.37
Mish	99.12±0.11	91.12±0.21	93.30±0.21
GELU	99.15±0.10	91.17±0.20	93.22±0.21
PReLU	99.01±0.17	90.89±0.25	93.05±0.28
ReLU6	99.07±0.10	90.99±0.24	93.10±0.20
PAU	99.14±0.09	91.20±0.19	93.17±0.20
ErfAct	**99.30**±0.08	**91.37**±0.20	**93.52**±0.21
Pserf	**99.32**±0.08	**91.31**±0.22	**93.43**±0.21

Table 5.4: Comparison between different baseline activations and ErfAct and Pserf on MNIST, Fashion MNIST, and SVHN datasets with LeNet model. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

(1998)), AlexNet (AN) (Krizhevsky *et al.* (2012)), PreActResnet-18 (PARN-18) (He *et al.* (2016)), Deep Layer Aggregation (DLA) (Yu *et al.* (2019)), Googlenet (GN) (Szegedy *et al.* (2014*a*)), Resnext-50 (Rxt) (Xie *et al.* (2017)), Xception (Xpt) (Chollet (2017)), ShuffleNet V2 (SN-V2) (Ma *et al.* (2018)), ResNet18 (RN-18) (He *et al.* (2015*a*)), and Network in Network (NIN) (Lin *et al.* (2014)) is reported in the Table 5.7 and 5.8. respectively. We get good improvement with EfficientNet B0, PreActResnet-18, LeNet, GoogleNet, Resnext-50, and ShuffleNet V2 models on both the datasets compared to ReLU or other baseline activation functions. From all the tables it is evident that the training is stable (mean$\pm$std) and the proposed activations archive 1%-6% higher Top-1 accuracy in most of models compared to the baselines. The networks are trained upto 200 epochs with SGD optimizer (Robbins and Monro (1951); Kiefer and Wolfowitz (1952)), 0.9 momentum, and $5e^{-4}$ weight decay. We have started with 0.01 initial learning rate and decay the learning rate with cosine annealing (Loshchilov and Hutter (2017)) learning rate scheduler. We consider batch size of 128. We consider standard data augmentation methods like horizontal flip, rotation for both the datasets. The Figures 5.4 and 5.5 shows the learning curves on CIFAR100 dataset with Shufflenet V2 (2.0x) model for the baseline and the proposed activation functions and it is noticeable that training & test accuracy curve is higher and loss curve is lower respectively for ErfAct and Pserf compared to the baseline activations.

We report more detailed results with Mixup (Zhang *et al.* (2017*a*)) augmentation method with ShuffleNet V2 (2.0x) and ResNet-18 models in Table 5.9. The table shows that the proposed activations beat the baseline activation functions in both models with Mixup augmentation. We consider the same experimental setup for Mixup as reported in the CIFAR section.

Tiny Imagenet:

We consider a more challenging and important classification dataset Tiny Imagenet (Le and Yang (2015)) which is a similar type of dataset like ILSVRC and consisting of 200 classes with RGB images of size 64×64 with total 1,00,000 training images, 10,000 validation images, and 10,000 test images. To compare the performance, we have considered WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)) model and Top-1 accuracy is reported in table 5.10 for mean of 5 different runs. The

Activation Function	VGG-16	WRN 28-10	ResNet-50	PA-ResNet-34	DN-121	IN-V3	MN-V2	SF-V2 2.0x
ReLU	71.67 ±0.28	76.32 ±0.25	74.17 ±0.24	73.12 ±0.23	75.67 ±0.28	74.23 ±0.26	74.02 ±0.24	67.49 ±0.26
Leaky ReLU	71.77 ±0.30	76.69 ±0.27	74.11 ±0.27	73.41 ±0.26	75.90 ±0.27	74.40 ±0.28	74.17 ±0.24	67.71 ±0.27
ELU	71.71 ±0.28	76.39 ±0.28	74.51 ±0.24	73.61 ±0.25	75.87 ±0.26	74.71 ±0.26	74.29 ±0.22	67.91 ±0.30
Swish	72.07 ±0.26	77.18 ±0.23	75.10 ±0.24	73.97 ±0.23	76.59 ±0.28	75.31 ±0.27	75.02 ±0.24	70.49 ±0.23
Softplus	71.10 ±0.32	75.36 ±0.37	74.19 ±0.38	73.17 ±0.36	75.08 ±0.36	74.20 ±0.34	74.33 ±0.38	68.93 ±0.36
Mish	72.31 ±0.24	77.40 ±0.25	76.30 ±0.22	75.14 ±0.21	77.11 ±0.25	76.22 ±0.25	75.31 ±0.21	71.79 ±0.22
GELU	71.98 ±0.25	77.35 ±0.25	75.61 ±0.22	74.28 ±0.23	76.79 ±0.27	75.52 ±0.25	75.21 ±0.23	70.35 ±0.27
PAU	71.72 ±0.25	77.20 ±0.26	75.89 ±0.24	74.41 ±0.23	76.59 ±0.28	75.79 ±0.28	75.07 ±0.19	70.68 ±0.26
PReLU	71.77 ±0.30	76.79 ±0.27	74.45 ±0.29	73.32 ±0.27	76.19 ±0.30	74.51 ±0.29	74.31 ±0.32	68.35 ±0.30
ReLU6	72.07 ±0.27	76.62 ±0.28	74.37 ±0.24	73.50 ±0.24	76.07 ±0.26	74.69 ±0.25	74.64 ±0.24	67.93 ±0.26
ErfAct	**72.93** ±0.22	**78.49** ±0.23	**77.09** ±0.20	**76.21** ±0.20	**78.18** ±0.23	**77.12** ±0.24	**76.23** ±0.19	**73.17** ±0.22
Pserf	**72.69** ±0.24	**78.31** ±0.24	**76.97** ±0.20	**75.91** ±0.22	**78.38** ±0.22	**77.01** ±0.25	**76.07** ±0.21	**72.91** ±0.21

Table 5.5: Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table.

Activation Function	VGG-16	WRN 28-10	ResNet-50	PA-ResNet-34	DN-121	IN-V3	MN-V2	SF-V2 2.0x
ReLU	93.44 ±0.22	95.17 ±0.21	94.35 ±0.18	94.17 ±0.19	94.77 ±0.20	94.15 ±0.20	94.20 ±0.16	91.63 ±0.21
Leaky ReLU	93.65 ±0.21	95.02 ±0.22	94.45 ±0.20	94.33 ±0.18	94.89 ±0.22	94.20 ±0.22	94.32 ±0.19	91.82 ±0.23
ELU	93.70 ±0.19	95.28 ±0.20	94.27 ±0.24	94.30 ±0.25	94.64 ±0.18	94.38 ±0.17	94.27 ±0.18	91.99 ±0.20
Swish	93.77 ±0.18	95.41 ±0.17	94.61 ±0.24	94.47 ±0.25	94.81 ±0.19	94.51 ±0.17	94.40 ±0.20	92.17 ±0.25
Softplus	93.10 ±0.33	94.77 ±0.30	93.91 ±0.30	94.07 ±0.35	94.41 ±0.34	94.21 ±0.32	93.79 ±0.29	91.32 ±0.33
Mish	93.91 ±0.17	95.35 ±0.18	94.78 ±0.22	94.55 ±0.23	95.03 ±0.15	94.64 ±0.18	94.71 ±0.18	92.41 ±0.20
GELU	93.71 ±0.17	95.28 ±0.19	94.64 ±0.23	94.31 ±0.25	94.99 ±0.19	94.57 ±0.21	94.40 ±0.18	92.27 ±0.20
PAU	93.57 ±0.22	95.27 ±0.20	94.67 ±0.23	94.41 ±0.24	94.74 ±0.20	94.57 ±0.19	94.51 ±0.14	92.30 ±0.21
PReLU	93.41 ±0.23	95.02 ±0.24	94.27 ±0.26	94.30 ±0.26	94.51 ±0.24	94.49 ±0.22	94.32 ±0.23	91.80 ±0.25
ReLU6	93.72 ±0.17	95.32 ±0.19	94.30 ±0.24	94.21 ±0.24	94.61 ±0.20	94.42 ±0.20	94.18 ±0.19	91.71 ±0.21
ErfAct	**94.47** ±0.15	**95.88** ±0.12	**95.01** ±0.17	**95.21** ±0.18	**95.71** ±0.15	**95.29** ±0.14	**95.34** ±0.12	**93.74** ±0.18
Pserf	**94.24** ±0.16	**95.71** ±0.13	**95.14** ±0.19	**95.08** ±0.29	**95.62** ±0.17	**95.10** ±0.13	**95.19** ±0.14	**93.59** ±0.18

Table 5.6: Comparison between different baseline activations and ErfAct and Pserf on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table.

Activation Function	EN-B0	LN	AN	PARN-18	DLA	GN	Rxt	Xpt	SN-V1	RN-18	NIN
ReLU	95.04 ±0.16	75.68 ±0.21	84.18 ±0.21	93.47 ±0.22	93.90 ±0.18	93.02 ±0.20	93.28 ±0.18	90.64 ±0.22	94.20 ±0.20	94.01 ±0.21	90.49 ±0.24
Leaky ReLU (α = 0.01)	95.22 ±0.16	75.91 ±0.22	84.32 ±0.23	93.61 ±0.21	94.01 ±0.20	92.91 ±0.18	93.39 ±0.19	90.80 ±0.24	94.32 ±0.22	94.12 ±0.24	90.59 ±0.26
ELU	95.35 ±0.18	76.10 ±0.20	84.78 ±0.20	93.65 ±0.22	93.96 ±0.20	93.06 ±0.17	93.55 ±0.23	91.38 ±0.24	94.32 ±0.21	94.19 ±0.24	90.55 ±0.24
Swish	95.60 ±0.17	77.55 ±0.19	85.10 ±0.20	93.87 ±0.20	94.25 ±0.17	93.30 ±0.20	93.69 ±0.19	91.94 ±0.20	94.65 ±0.19	94.29 ±0.21	90.97 ±0.25
Softplus	95.10 ±0.27	75.65 ±0.33	84.22 ±0.30	93.12 ±0.26	93.71 ±0.25	92.64 ±0.29	93.01 ±0.29	90.69 ±0.30	93.92 ±0.25	93.99 ±0.27	90.39 ±0.30
Mish	95.75 ±0.15	**78.76** ±0.16	85.70 ±0.18	93.70 ±0.24	94.40 ±0.17	93.22 ±0.20	93.92 ±0.17	92.15 ±0.19	94.78 ±0.19	94.45 ±0.25	**91.17** ±0.23
GELU	95.39 ±0.19	77.79 ±0.17	85.15 ±0.21	93.77 ±0.19	94.12 ±0.20	93.45 ±0.20	93.77 ±0.19	91.89 ±0.22	94.55 ±0.21	94.45 ±0.24	91.01 ±0.24
PReLU	95.20 ±0.18	75.85 ±0.24	84.38 ±0.23	93.46 ±0.25	93.01 ±0.22	92.89 ±0.24	93.45 ±0.26	91.29 ±0.26	94.34 ±0.22	94.15 ±0.28	90.83 ±0.27
ReLU6	95.43 ±0.16	75.71 ±0.18	84.64 ±0.22	93.75 ±0.22	94.09 ±0.17	92.89 ±0.18	93.48 ±0.22	91.38 ±0.24	94.22 ±0.20	94.28 ±0.24	90.87 ±0.24
PAU	95.35 ±0.17	77.59 ±0.21	85.01 ±0.24	93.75 ±0.22	94.34 ±0.20	93.29 ±0.21	93.52 ±0.20	91.79 ±0.21	94.60 ±0.21	94.31 ±0.21	90.97 ±0.25
ErfAct	**96.10** ±0.15	77.48 ±0.19	**87.01** ±0.20	**94.10** ±0.20	**94.67** ±0.17	**94.12** ±0.18	**94.17** ±0.18	**93.01** ±0.17	**95.14** ±0.19	**94.71** ±0.23	90.81 ±0.24
Pserf	**95.98** ±0.18	77.52 ±0.20	**87.15** ±0.20	**94.01** ±0.22	**94.56** ±0.20	**93.95** ±0.20	**94.01** ±0.19	**93.18** ±0.18	**94.96** ±0.18	**94.68** ±0.18	90.78 ±0.24

Table 5.7: Comparison between different baseline activations and ErfAct and Pserf on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table.

Activation Function	EN-B0	LN	AN	PARN-18	DLA	GN	Rxt	Xpt	SN-V1	RN-18	NIN
ReLU	76.45 ±0.26	45.50 ±0.30	55.02 ±0.30	73.10 ±0.20	74.50 ±0.22	72.64 ±0.28	74.31 ±0.22	71.20 ±0.20	73.70 ±0.23	73.17 ±0.25	65.12 ±0.25
Leaky ReLU (α = 0.01)	76.70 ±0.25	45.64 ±0.28	55.34 ±0.28	73.30 ±0.21	74.62 ±0.23	72.51 ±0.28	74.60 ±0.23	71.10 ±0.24	73.89 ±0.25	73.21 ±0.23	65.27 ±0.23
ELU	76.77 ±0.26	45.23 ±0.27	55.72 ±0.28	73.41 ±0.23	74.54 ±0.24	72.85 ±0.27	74.71 ±0.24	71.40 ±0.22	73.98 ±0.21	73.40 ±0.25	65.39 ±0.23
Swish	77.34 ±0.20	47.30 ±0.25	57.64 ±0.28	74.98 ±0.24	75.20 ±0.20	73.45 ±0.28	75.06 ±0.26	72.16 ±0.24	74.29 ±0.22	73.65 ±0.24	66.20 ±0.22
Softplus	76.41 ±0.30	44.10 ±0.38	54.85 ±0.36	73.10 ±0.35	74.31 ±0.26	72.09 ±0.35	74.20 ±0.34	71.51 ±0.36	73.90 ±0.27	72.80 ±0.36	65.25 ±0.30
Mish	78.02 ±0.23	**47.49** ±0.28	58.35 ±0.25	74.84 ±0.24	75.45 ±0.20	73.85 ±0.25	76.07 ±0.24	73.34 ±0.23	74.40 ±0.21	74.39 ±0.22	**66.50** ±0.22
GELU	77.30 ±0.24	47.23 ±0.25	57.55 ±0.27	74.87 ±0.23	75.20 ±0.23	73.32 ±0.27	75.32 ±0.23	72.25 ±0.22	73.15 ±0.22	73.77 ±0.22	66.01 ±0.22
PReLU	76.62 ±0.28	45.69 ±0.30	55.41 ±0.30	73.16 ±0.25	74.98 ±0.24	72.60 ±0.30	74.50 ±0.26	71.30 ±0.23	73.79 ±0.24	73.10 ±0.26	65.56 ±0.27
ReLU6	76.58 ±0.23	45.86 ±0.28	55.75 ±0.28	73.30 ±0.25	74.69 ±0.21	72.40 ±0.24	74.69 ±0.24	71.40 ±0.24	73.99 ±0.23	73.30 ±0.25	65.42 ±0.24
PAU	77.21 ±0.26	47.17 ±0.28	57.42 ±0.27	74.71 ±0.22	75.50 ±0.22	73.60 ±0.28	75.60 ±0.25	72.50 ±0.24	74.36 ±0.22	73.99 ±0.22	66.20 ±0.22
ErfAct	**78.97** ±0.23	47.29 ±0.26	**60.89** ±0.25	**75.77** ±0.24	**76.43** ±0.18	**74.47** ±0.26	**77.23** ±0.23	**74.32** ±0.22	**74.90** ±0.21	**74.79** ±0.24	66.25 ±0.22
Pserf	**78.75** ±0.24	47.27 ±0.27	**60.57** ±0.24	**75.60** ±0.25	**76.23** ±0.20	**74.50** ±0.25	**77.10** ±0.24	**74.20** ±0.24	**74.72** ±0.20	**74.84** ±0.23	66.35 ±0.23

Table 5.8: Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 12 different runs have been reported. mean±std is reported in the table.

Activation Function	ShuffleNet V2 (2.0x)	ResNet 18
ReLU	70.02 ± 0.22	73.72 ± 0.23
Leaky ReLU	69.85 ± 0.24	73.91 ± 0.24
ELU	70.25 ± 0.23	73.92 ± 0.26
Swish	73.12 ± 0.23	74.52 ± 0.23
Softplus	69.52 ± 0.30	73.63 ± 0.26
Mish	73.65 ± 0.21	74.97 ± 0.24
GELU	73.25 ± 0.23	74.45 ± 0.23
PReLU	70.05 ± 0.23	74.10 ± 0.27
ReLU6	70.20 ± 0.24	74.01 ± 0.24
PAU	73.28 ± 0.24	74.65 ± 0.25
ErfAct	**75.07** ± 0.22	**75.67** ± 0.21
Pserf	**74.84** ± 0.24	**75.46** ± 0.21

Table 5.9: Comparison between different baseline activations and ErfAct and Pserf on CIFAR100 dataset. Top-1 accuracy(in %) with Mixup augmentation method for mean of 12 different runs have been reported. mean±std is reported in the table.

model is trained with a batch size of 32, He Normal initializer (He *et al.* (2015*b*)), 0.2 dropout rate (Srivastava *et al.* (2014)), adam optimizer (Kingma and Ba (2015)), with initial learning rate(lr rate) 0.01, and lr rate is reduced by a factor of 10 after every 60 epochs up-to 300 epochs. We have considered the standard data augmentation methods like rotation, width shift, height shift, shearing, zoom, horizontal flip, fill mode. From the table, it is clear that the performance for the proposed functions are better than the baseline functions and stable (mean±std) and got a boost in Top-1 accuracy by 2.59% and 2.40% for ErfAct and Pserf compared to ReLU.

ImageNet-1k

ImageNet-1k (Deng *et al.* (2009)) is a widely used computer vision database with more than 1.2 million training images and have 1000 different classes. We report result with ShuffleNet V2 (1.0x) (Ma *et al.* (2018)) model on ImageNet-1k dataset in Table 5.11. We use SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)), 0.9 momentum, $5e^{-4}$ weight decay, and a batch size of 256 and trained upto 600k iterations. Experiments on ImageNet-1k is conducted on four NVIDIA V100 GPUs with 32GB RAM each.

Activation Function	Wide ResNet 28-10 Model
ReLU	61.61 $\pm$ 0.47
Swish	62.44 $\pm$ 0.49
Leaky ReLU	61.47 $\pm$ 0.44
ELU	61.99 $\pm$ 0.57
Softplus	60.42 $\pm$ 0.61
Mish	63.02 $\pm$ 0.57
GELU	62.64 $\pm$ 0.62
PAU	62.04 $\pm$ 0.54
PReLU	61.25 $\pm$ 0.51
ReLU6	61.72 $\pm$ 0.56
ErfAct	**64.20** $\pm$ 0.51
Pserf	**64.01** $\pm$ 0.49

Table 5.10: Comparison between different baseline activations and ErfAct and Pserf on Tiny ImageNet dataset. Mean of 5 different runs for Top-1 accuracy(in $\%$) have been reported. mean$\pm$std is reported in the table.

Activation Function	ShuffleNet V2 (1.0x)
ReLU	69.20
Leaky ReLU	69.32
PReLU	69.28
ReLU6	69.40
ELU	69.24
Softplus	69.07
Swish	70.06
GELU	69.91
Mish	69.95
PAU	70.17
ErfAct	**70.65**
Pserf	**70.57**

Table 5.11: Top-1 Accuracy reported on ImageNet-1k dataset.

5.5.2 Semantic Segmentation

Semantic segmentation is an important problem in deep learning. In this section, we present experimental results on the Cityscapes dataset (Cordts *et al.* (2016)). We report the pixel accuracy and mean Intersection-Over-Union (mIOU) on the U-net model (Ronneberger *et al.* (2015)). The model is trained up to 250 epochs, with adam optimizer (Kingma and Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)). A mean of 5 different runs on the test dataset is reported in table 5.12. We got around 1.97% and 1.89% boost on mIOU for ErfAct and Pserf compared to ReLU.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.60 $\pm$ 0.45	69.32 $\pm$ 0.30
Swish	79.71 $\pm$ 0.49	69.68 $\pm$ 0.31
Leaky ReLU	79.41 $\pm$ 0.42	69.48 $\pm$ 0.39
ELU	79.27 $\pm$ 0.54	68.12 $\pm$ 0.41
Softplus	78.69 $\pm$ 0.49	68.12 $\pm$ 0.55
Mish	80.12 $\pm$ 0.45	69.87 $\pm$ 0.29
GELU	79.60 $\pm$0.39	69.51 $\pm$ 0.39
PAU	79.95 $\pm$ 0.41	69.42 $\pm$ 0.46
PReLU	78.99 $\pm$ 0.42	68.82 $\pm$ 0.41
ReLU6	79.59 $\pm$ 0.41	69.66 $\pm$ 0.41
ErfAct	**81.41** $\pm$ 0.45	**71.29** $\pm$ 0.31
Pserf	**81.12** $\pm$ 0.42	**71.21** $\pm$ 0.34

Table 5.12: Comparison between different baseline activations and ErfAct and Pserf on semantic segmentation problem on U-NET model in CityScapes dataset. mean$\pm$std is reported in the table.

5.5.3 Object Detection

Object detection is a standard problem in computer vision. In this section, we have reported our experimental results on challenging Pascal VOC dataset (Everingham *et al.* (2010)) with Single Shot MultiBox Detector(SSD) 300 (Liu *et al.* (2016)) with VGG-16(with batch-normalization) (Simonyan and Zisserman (2015)) as the backbone network. The mean average precision (mAP) is reported in Table 5.13 for a mean of 8 different runs. The model is trained with batch size of 8, 0.001 learning rate, SGD optimizer (Robbins and Monro (1951); Kiefer and Wolfowitz (1952)) with 0.9 momen-

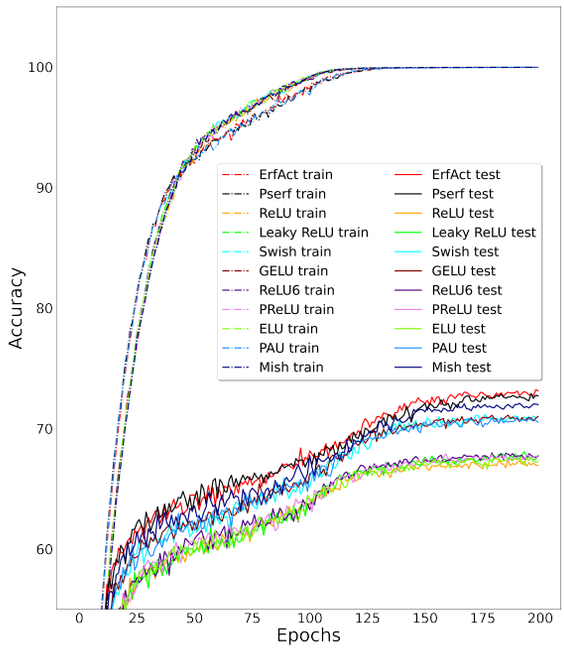

Figure 5.4: Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations, ErfAct, and Pserf.

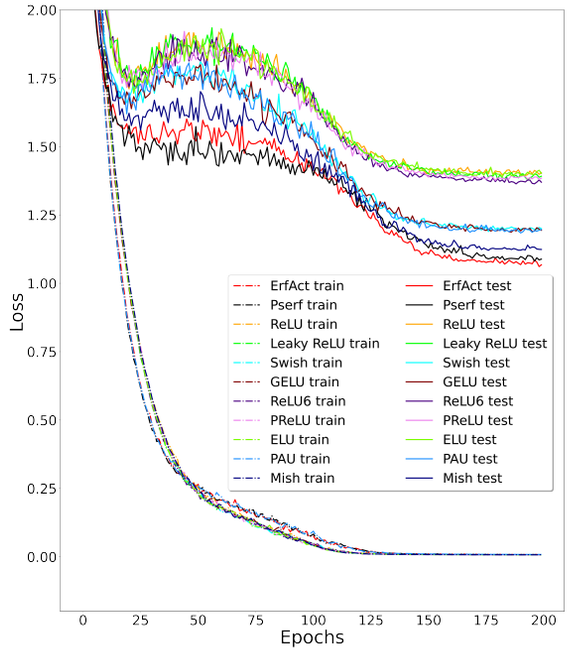

Figure 5.5: Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations, ErfAct, and Pserf.

tum, $5e^{-4}$ weight decay for 120000 iterations. The results are stable on different runs (mean$\pm$std). We got around 1% boost in mAP for both ErfAct and Pserf compared to ReLU.

Activation Function	mAP
ReLU	77.2 ± 0.14
Swish	77.5 ± 0.12
Leaky ReLU	77.2 ± 0.19
ELU	75.1 ± 0.22
Softplus	74.2 ± 0.25
Mish	77.6 ± 0.14
GELU	77.5 ± 0.14
PAU	77.4 ± 0.16
PReLU	77.2 ± 0.20
ReLU6	77.1 ± 0.15
ErfAct	$\mathbf{78.2} \pm 0.12$
Pserf	$\mathbf{78.2} \pm 0.14$

Table 5.13: Comparison between different baseline activations and ErfAct and Pserf on Object Detection problem on SSD 300 model in Pascal-VOC dataset. mean$\pm$std is reported in the table.

.

5.5.4 Machine Translation

Activation Function	BLEU Score on the newstest2014 dataset
ReLU	26.2 $\pm$ 0.15
Swish	26.4 $\pm$ 0.10
Leaky ReLU	26.3 $\pm$ 0.17
ELU	25.1 $\pm$ 0.15
Softplus	23.6 $\pm$ 0.16
Mish	26.3 $\pm$ 0.12
GELU	26.4 $\pm$ 0.19
PAU	26.3 $\pm$ 0.16
PReLU	26.2 $\pm$ 0.21
ReLU6	26.1 $\pm$ 0.14
ErfAct	**26.8** $\pm$ 0.11
Pserf	**26.7** $\pm$ 0.10

Table 5.14: Comparison between different baseline activations and ErfAct and Pserf on Machine translation problem on transformer model in WMT-2014 dataset. mean$\pm$std is reported in the table.

Machine Translation is a procedure in which text or speech is translated from one language to another language without the help of any human being. We consider the standard WMT 2014 English$\rightarrow$German dataset for our experiment. The database contains 4.5 million training sentences. We train an attention-based 8-head transformer network (Vaswani *et al.* (2017)) with Adam optimizer (Kingma and Ba (2015)), 0.1 dropout rate (Srivastava *et al.* (2014)), and train up to 100000 steps. We try to keep other hyperparameters similar as mentioned in the original paper (Vaswani *et al.* (2017)). We evaluate the network performance on the newstest2014 dataset using the BLEU score metric. The mean of 5 different runs is being reported on Table 5.14 on the test dataset(newstest2014). From the table, it is clear that the results are stable on different runs (mean$\pm$std), and we got around 0.6% and 0.5% boost in BLEU score for ErfAct and Pserf compared to ReLU.

5.6 Baseline Table

The experiment section shows that ErfAct and Pserf beat or perform equally well with baseline activation functions in most cases while under-performs marginally on rare

Baselines	ReLU	Leaky ReLU	ELU	Softplus	Swish	PReLU	ReLU6	Mish	GELU	PAU
ErfAct > Baseline	57	57	57	57	54	56	56	53	55	55
ErfAct = Baseline	0	0	0	0	0	0	0	0	0	0
ErfAct < Baseline	0	0	0	0	3	1	1	4	2	2
Pserf > Baseline	57	57	57	57	54	56	56	53	55	54
Pserf = Baseline	0	0	0	0	0	0	0	0	0	0
Pserf < Baseline	0	0	0	0	3	1	1	4	2	3

Table 5.15: Baseline table for ErfAct and Pserf. These numbers represent the total number of models in which ErfAct and Pserf underperforms, equal or outperforms compared to the baseline activation functions

occasions. We provide a detailed comparison based on all the experiments in earlier sections with the proposed and the baseline activation functions in Table 5.15.

5.7 Computational Time Comparison

Activation Function	Forward Pass	Backward Pass
ReLU	5.39 ± 0.39 μs	5.70 ± 1.56 μs
Swish	8.35 ± 1.44 μs	10.56 ± 2.34 μs
Leaky ReLU	5.50 ± 0.51 μs	5.97 ± 0.75 μs
ELU	6.17 ± 0.50 μs	5.93 ± 0.93 μs
Softplus	6.13 ± 0.49 μs	5.94 ± 0.55 μs
Mish	7.45 ± 2.55 μs	8.89 ± 2.85 μs
GELU	8.87 ± 1.54 μs	9.22 ± 1.75 μs
PAU	19.05 ± 2.69 μs	32.62 ± 3.76 μs
PReLU	6.12 ± 0.90 μs	6.23 ± 1.41 μs
ReLU6	5.77 ± 0.73 μs	5.73 ± 0.66 μs
ErfAct	7.41 ± 1.51 μs	10.62 ± 1.53 μs
Pserf	7.53 ± 1.77 μs	10.77 ± 1.78 μs

Table 5.16: Runtime comparison for the forward and backward passes for ErfAct and Pserf and baseline activation functions for a $32\times$ 32 RGB image in PreActResNet-18 model.

We present the time comparison for the baseline activation functions and ErfAct, Pserf for the mean of 100 runs for both forward and backward pass on a 32×32 RGB image in PreActResNet-18 He *et al.* (2016) model in Table 5.16. An NVIDIA Tesla V100 GPU with 32GB ram is used to run the experiments. From Table 5.16 and the experiment section, it is clear that there is a small trade-off between the computational time and the model performance when compared to ReLU as the proposed activations

contain trainable parameters. In contrast, the time is comparable with Swish, Mish or GELU & much better than PAU and model performance comparatively much better than baseline activations in most cases.

5.8 Conclusion

In this chapter, two simple and effective novel activation functions have been proposed. They are named as ErfAct and Pserf. The proposed functions are unbounded above, bounded below, non-monotonic, smooth and zero-centred. It is shown that both functions can approximate the ReLU activation function. Both functions have similar training times, like Swish, GELU, and Mish. Across most of the experiments, ErfAct and Pserf are top-performing activation functions from which it can be concluded that the proposed functions have the potential to replace the widely used activations like ReLU, Swish or Mish.

CHAPTER 6

Maximum Activation Unit

6.1 Introduction

Deep Neural network has emerged a lot in recent years and has significantly impacted our real-life applications. Neural networks are the backbone of deep learning. An activation function is the brain of the neural network, which plays a central role in the effectiveness & training dynamics of deep neural networks. Hand-designed activation functions are quite a common choice in neural network models. ReLU (Nair and Hinton (2010)) is a widely used hand-designed activation function. Despite its simplicity, ReLU has a significant drawback, known as the dying ReLU problem, in which up to 50% of neurons can be dead during network training. To overcome the shortcomings of ReLU, many activations have been proposed in recent years. Leaky ReLU (Maas *et al.* (2013*a*)), Parametric ReLU (He *et al.* (2015*b*)), ELU (Clevert *et al.* (2016)), Softplus (Zheng *et al.* (2015)), Randomized Leaky ReLU (Xu *et al.* (2015*a*)) are a few of them though they marginally improve performance of ReLU. Swish (Ramachandran *et al.* (2017)) is a non-linear activation function proposed by the Google brain team, showing some good improvement of ReLU. GELU (Hendrycks and Gimpel (2020)) is another popular smooth activation function. It can be shown that Swish and GELU are both are approximation by a smooth function of ReLU. Recently, a few non-linear activations have been proposed that improve the performance of ReLU, Swish or GELU. Some of them are either hand-designed or approximation by a smooth function of the Leaky ReLU function, Swish (Ramachandran *et al.* (2017)), Mish Misra (2020), and Padé activation unit (Molina *et al.* (2020)) are a few of them.

6.2 Related works and Motivation

The deep learning community is quite interested in proposing hand-designed activation functions. ReLU (Nair and Hinton (2010)), Leaky ReLU (Maas *et al.* (2013*a*)),

Parametric ReLU (He *et al.* (2015*b*)), ReLU6 (Krizhevsky (2010)), ELU (Clevert *et al.* (2016)), Swish (Ramachandran *et al.* (2017), and Softplus (Zheng *et al.* (2015)) are hand-designed widely used activation functions. While ReLU is the first choice for the deep learning community due to its simplicity, it has some serious drawbacks. To overcome it, some better activations have been proposed so far. Swish, Softplus, ELU, and Leaky ReLU are a few of them. At the same time, only a few activation functions have been proposed so far by the direct approximation method, which is approximation by a smooth function of the Leaky ReLU or maximum function. Padé Activation Unit (Molina *et al.* (2020)) is proposed by approximating the Leaky ReLU function, and it performs better than ReLU, Swish in the image classification problem.

6.3 Research Contribution

In the last chapter, two hand-designed smooth activation functions have been proposed, which are approximation by a smooth function of the ReLU function. That work has been extended in this chapter. Instead of approximating only the ReLU function, the proposed functions approximate the maximum function using ErfAct, Pserf, and parametric Mish. The proposed functional form is called Maximum Activation Unit (MAU). The extensive experiments presented in the experiment section on four important deep learning problems (image classification, object detection, semantic segmentation, and machine translation) in widely used standard datasets with a large number of architectures prove the efficacy of the proposed activation functions.

6.4 Maximum Activation Unit

In this chapter, I am interested in looking at the maximum function, like the SMU chapter. The maximum function is not differentiable at the origin and can be approximated by smooth functions, as explored in the SMU chapter. Instead of approximating the maximum function by previously proposed functions directly. Note that I can rewrite

the maximum function in the following way:

$$max(x_1, x_2) = \begin{cases} x_1 & \text{if } x_1 \geq x_2 \\ x_2 & \text{otherwise} \end{cases}$$
$$= x_1 + max(0, x_2 - x_1) \tag{6.1}$$

Recall that the maximum function is not differentiable at the origin as explored in the SMU chapter. An interesting consequence of rewriting the maximum function in this way is the observation that the second summoned in the above equation actually looks like the ReLU function. Since I already have a good theory or approximation of the ReLU function (as explored in the previous chapter), using differentiable functions, I have used such approximations here to approximate this portion of the maximum function. Then I add x_1 to get back an approximation of the maximum function. Complicated cases can be constructed by considering nonlinear choices of x_1 and x_2. But, for rest of the chapter, I will only focus on the Leaky ReLU approximation of the equation 6.1. In the experimental section, I run extensive experiments to explore whether this significantly impacts training and testing accuracy. I found that the proposed functions actually perform better than the traditional functions like ReLU, Swish etc. I can approximate the maximum function using approximation by a smooth function. From equation 6.1, an approximation by a smooth function of the maximum function can be derived. I will only consider approximations of ReLU or Leaky ReLU functions to keep everything simple. From the last chapter, it is known that the ErfAct (Biswas *et al.* (2021*c*)), Pserf (Biswas *et al.* (2021*c*)), and parametric form of Mish (Misra (2020)) are approximation by a smooth function of ReLU. From equation 6.1, the formula for Leaky ReLU can be written as follows:

$$max(\alpha x, x) = \alpha x + max(0, (1 - \alpha)x) \tag{6.2}$$

ErfAct is defined as

$$f(x; \beta, \gamma) = x \operatorname{erf}(\beta e^{\gamma x})) \tag{6.3}$$

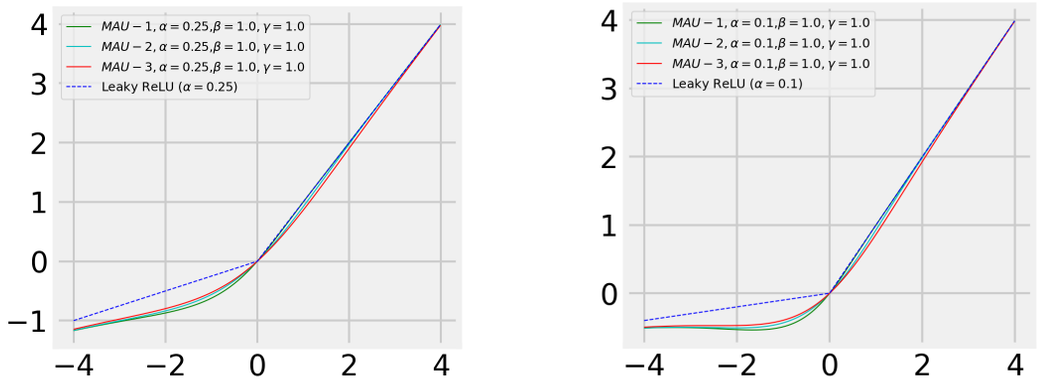

Figure 6.1: Approximation of Leaky ReLU ($\alpha = 0.25$) and Leaky ReLU ($\alpha = 0.1$) using MAU.

where erf is the Gaussian error function defined as follows:

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2}\, dt. \tag{6.4}$$

Notice that, for any fixed β, as $\gamma \to \infty$, ErfAct converges to ReLU smoothly. Similarly, it can be shown that Pserf and parametric Mish are approximation by a smooth function of ReLU. Replacing 2nd term in 6.2 by 6.3, we have approximation by a smooth function of Leaky ReLU or PReLU (depending on α, β, γ are hyperparameters or trainable parameters) as

$$max(\alpha x, x; \beta, \gamma) \approx F_1(\alpha x, x; \beta, \gamma) = \alpha x + (1-\alpha)x\, \text{erf}(\beta e^{(1-\alpha)\gamma x})) \tag{6.5}$$

Similarly, for Pserf and Parametric Mish, we have

$$max(\alpha x, x; \beta, \gamma) \approx F_2(\alpha x, x; \beta, \gamma) = \alpha x + (1-\alpha)x\, \text{erf}(\beta ln(1 + e^{(1-\alpha)\gamma x}))) \tag{6.6}$$

and

$$max(\alpha x, x; \beta, \gamma) \approx F_3(\alpha x, x; \beta, \gamma) = \alpha x + (1-\alpha)x\, \tanh(\beta ln(1 + e^{(1-\alpha)\gamma x}))) \tag{6.7}$$

For the rest of this chapter, I will call these functions in equation 6.5, 6.6, and 6.7 as MAU-1, MAU-2, and MAU-3 respectively. Observe that as $\gamma \to \infty$, these three functions will smoothly approximate ReLU or Leaky ReLU (depending on the value of α).

The corresponding derivative of the equation (6.5) for input variable x is

$$\begin{aligned}\frac{d}{dx}F_1(x,\alpha x,x;\beta,\gamma) = [(\alpha+(1-\alpha)\ \text{erf}(\beta e^{(1-\alpha)\gamma x}))\\ +\frac{2}{\sqrt{\pi}}\beta\gamma(1-\alpha)^2 x e^{(1-\alpha)\gamma x}e^{-(\beta e^{(1-\alpha)\gamma x})^2}\end{aligned} \tag{6.8}$$

where $\frac{d}{dx}\text{erf}(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}$.

Similarly, the derivatives of equation (6.6) and equation (6.7) can be derived for input variable x. A plot of MAU-1, MAU-2, and MAU-3 are given in figure 6.1.

6.4.1 Learning activation parameters via back-propagation

The trainable parameters are updated via backpropagation (LeCun *et al.* (1989)) algorithm. I have considered Pytorch (Paszke *et al.* (2019)) & Tensorflow-Keras (Chollet *et al.* (2015)) API to implement Forward and backward pass, and parameters are updated via automatic differentiation methods. Alternatively, CUDA (Nickolls *et al.* (2008)) based implementation (see (Maas *et al.* (2013*a*))) can be used to update the trainable parameters. the For a single layer, the gradient of a parameter Θ is:

$$\frac{\partial L}{\partial \Theta} = \sum_x \frac{\partial L}{\partial f(x)}\frac{\partial f(x)}{\partial \Theta} \tag{6.9}$$

where L is the objective function, $\Theta \in \{\alpha,\beta,\gamma\}$ and $f(x) \in \{F_1(\alpha x,x;\beta,\gamma), F_2(\alpha x,x;\beta,\gamma), F_3(\alpha x,x;\beta,\gamma)\}$. α,β, and γ can be used as trainable parameters or hyperparameters.
Now, note that the class of neural networks with MAU-1, MAU-2, and MAU-3 activation function is dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K.
The proof follows from the following proposition (see Molina *et al.* (2020)).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2020 Kidger and Lyons (2020)) **:-** Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

6.5 Experiments

To explore and compare the performance of MAU-1, MAU-2, and MAU-3, I have considered eight standard activation functions on different standard datasets and popular network architectures. I have considered four different deep learning problems like image classification, object detection, semantic segmentation, and machine translation. The following activation functions are considered as baseline functions to compare with the proposed activations: ReLU, ReLU6, Softplus, Leaky ReLU, Parametric ReLU (PReLU), ELU, PAU, Swish, and GELU. It is clear from the experimental results in the next sections that MAU-1, MAU-2, and MAU-3 outperform in most cases compared to the standard activations. I consider α as a hyperparameter and β & γ as trainable parameters for the rest of our experiments. All the experiments are conducted on an NVIDIA V100 GPU with 32GB RAM.

6.5.1 Image Classification

MNIST, Fashion MNIST and The Street View House Numbers (SVHN) Database:

In this section, I present results on MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), and SVHN (Netzer *et al.* (2011)) datasets. These three datasets have total 10 classes. I consider standard data augmentation methods like shearing, rotation, height shift, zoom on the three datasets. I report results for mean of 20 different runs with AlexNet (Krizhevsky *et al.* (2012)), and VGG-16 (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))) architecture in Table 6.1, Table 6.2 respectively. To train a network on these three datasets, I use a batch size of 128, stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, and trained all networks up-to 100 epochs. I consider 0.01 initial learning rate and decay the learning rate with cosine annealing (Loshchilov and Hutter (2017)) learning rate scheduler.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.52 ± 0.07	92.81 ± 0.19	95.14 ± 0.15
Leaky ReLU	99.48 ± 0.08	92.84 ± 0.22	95.26 ± 0.18
PReLU	99.50 ± 0.08	92.84 ± 0.20	95.18 ± 0.17
ReLU6	99.54 ± 0.06	93.08 ± 0.17	95.26 ± 0.14
ELU	99.54 ± 0.07	92.92 ± 0.17	95.35 ± 0.18
Softplus	99.19 ± 0.12	92.37 ± 0.24	94.92 ± 0.20
PAU	99.50 ± 0.07	93.07 ± 0.17	95.20 ± 0.14
Swish	99.60 ± 0.06	92.91 ± 0.17	$\mathbf{95.42} \pm 0.15$
GELU	99.54 ± 0.08	93.10 ± 0.16	95.40 ± 0.16
MAU-1	$\mathbf{99.65} \pm 0.05$	$\mathbf{93.22} \pm 0.13$	95.37 ± 0.10
MAU-2	$\mathbf{99.66} \pm 0.04$	$\mathbf{93.29} \pm 0.12$	95.39 ± 0.12
MAU-3	$\mathbf{99.63} \pm 0.05$	$\mathbf{93.32} \pm 0.13$	95.32 ± 0.11

Table 6.1: A Detailed Comparison between MAU-1, MAU-2, and MAu-3 and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with AlexNet Model. Top-1 Test Accuracy (in %) is reported for the Mean of 20 Different Runs. mean$\pm$std is reported in the Table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.53 ± 0.09	93.70 ± 0.15	96.12 ± 0.14
Leaky ReLU	99.55 ± 0.07	93.95 ± 0.16	96.14 ± 0.15
PReLU	99.59 ± 0.08	93.88 ± 0.15	96.11 ± 0.18
ReLU6	99.56 ± 0.07	93.90 ± 0.10	96.21 ± 0.17
ELU	99.50 ± 0.08	93.86 ± 0.18	96.16 ± 0.14
Softplus	99.31 ± 0.14	93.65 ± 0.20	95.85 ± 0.22
PAU	99.56 ± 0.07	94.29 ± 0.14	96.25 ± 0.16
Swish	99.53 ± 0.08	94.15 ± 0.14	96.30 ± 0.15
GELU	99.59 ± 0.06	94.20 ± 0.13	96.20 ± 0.14
MAU-1	$\mathbf{99.68} \pm 0.06$	$\mathbf{94.47} \pm 0.11$	$\mathbf{96.46} \pm 0.10$
MAU-2	$\mathbf{99.66} \pm 0.05$	$\mathbf{94.42} \pm 0.10$	$\mathbf{96.48} \pm 0.11$
MAU-3	$\mathbf{99.65} \pm 0.04$	$\mathbf{94.45} \pm 0.13$	$\mathbf{96.48} \pm 0.11$

Table 6.2: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on MNIST, Fashion MNIST, and SVHN Datasets for Image Classification Problem with VGG16 Model. Top-1 Test Accuracy (in %) for the Mean of 20 Different Runs is reported. mean$\pm$std is reported in the table.

CIFAR:

The CIFAR (Krizhevsky (2009)) is considered for image classification task and it has a total of 60k 32×32 RGB images and is divided into 50k training and 10k test images. CIFAR is divided into two different datasets- CIFAR10 and CIFAR100 with a total of 10 and 100 classes, respectively. Top-1 accuracy is reported on Table 6.3, 6.4, 6.5 on CIFAR10 dataset and on Table 6.6, 6.7, 6.8 on CIFAR100 dataset. I consider MobileNet V1 (Howard *et al.* (2017)), MobileNet V2 (Sandler *et al.* (2019)), ResNet (He *et al.* (2015*a*)), PreActResNet (He *et al.* (2016)), Shufflenet V2 (Ma *et al.* (2018)), Inception V3 (Szegedy *et al.* (2015*a*)), squeeze and excitation networks (SeNet) (Hu *et al.* (2017)), ResNext (Xie *et al.* (2017)), DenseNet (Huang *et al.* (2016*a*)), Xception (Chollet (2017)), Squeezenet (Iandola *et al.* (2016)), WideResNet (Zagoruyko and Komodakis (2016)), VGG (Simonyan and Zisserman (2015)) (with batch-normalization (Ioffe and Szegedy (2015))), AlexNet Krizhevsky *et al.* (2012), LeNet (Lecun *et al.* (1998)), and EfficientNet B0 (Tan and Le (2020)). Standard data augmentation methods like height shift, horizontal flip, width shift, and rotation is applied on both datasets. For all the experiments on CIFAR10 & CIFAR100 to train a network, I consider stochastic gradient descent (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) optimizer with 0.9 momentum & $5e^{-4}$ weight decay, a batch size of 128, and trained all networks up-to 200 epochs. I start with 0.01 learning rate and decay the learning rate by a factor of 10 after every 60 epochs. It is quite clear from these six tables that replacing ReLU by MAU-1, MAU-2, and MAU-3, there is an improvement in top-1 accuracy from 1% to more than 5% in most of the models. Accuracy and loss curves are given in Figure 6.2 & Figure 6.3 respectively. It is quite clear from the learning curves that after training few epochs, MAU-1, MAU-2, and MAU-3 has stable & smooth learning, and higher accuracy and lower loss on the test dataset compared to other baseline activation functions. I consider state of the art data augmentation method like Mixup (Zhang *et al.* (2017*a*)) on CIFAR 100 dataset with ShuffleNet V2 (2.0x), ResNet 18 & ResNet 50 models to compare the performance of MAU-1, MAU-2, and MAU-3 with the baseline activations. MAU-1, MAU-2, and MAU-3 got improvement over the baseline activation functions with Mixup augmentation method. Results are reported on Table 6.9 for the mean of 20 different runs. I consider the same experimental setup as I have used for the CIFAR100 dataset.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MobileNet V1	MobileNet V2	ResNet 18	ResNet 34	ResNet 50
ReLU	88.15 ±0.22	90.80 ±0.25	91.12 ±0.24	91.42 ±0.24	92.40 ±0.25	93.75 ±0.23	93.34 ±0.20	93.85 ±0.24	93.98 ±0.24
Leaky ReLU	88.20 ±0.23	90.92 ±0.27	91.00 ±0.22	91.52 ±0.23	92.55 ±0.27	93.82 ±0.20	93.20 ±0.21	93.78 ±0.23	93.71 ±0.23
PReLU	88.22 ±0.23	90.81 ±0.29	91.11 ±0.25	91.32 ±0.24	92.41 ±0.23	94.12 ±0.23	93.29 ±0.22	93.90 ±0.25	93.92 ±0.22
ReLU6	88.29 ±0.22	90.96 ±0.23	91.19 ±0.22	91.56 ±0.22	92.60 ±0.23	93.81 ±0.21	93.30 ±0.20	93.90 ±0.20	93.82 ±0.20
ELU	88.20 ±0.21	90.74 ±0.26	91.20 ±0.22	91.50 ±0.22	92.49 ±0.23	93.81 ±0.20	93.45 ±0.23	94.10 ±0.23	93.81 ±0.22
Softplus	87.81 ±0.27	90.59 ±0.30	91.07 ±0.26	91.47 ±0.22	92.40 ±0.30	93.94 ±0.22	93.23 ±0.23	93.50 ±0.26	93.75 ±0.26
PAU	88.90 ±0.22	91.62 ±0.25	92.43 ±0.20	92.83 ±0.24	92.78 ±0.17	94.79 ±0.20	93.74 ±0.19	94.20 ±0.22	94.22 ±0.15
Swish	88.70 ±0.22	91.80 ±0.23	92.18 ±0.21	92.51 ±0.21	92.41 ±0.20	94.70 ±0.20	93.72 ±0.19	94.15 ±0.22	94.30 ±0.17
GELU	88.52 ±0.20	91.89 ±0.21	92.55 ±0.20	92.87 ±0.24	92.30 ±0.22	94.67 ±0.22	**93.77** ±0.20	94.07 ±0.22	94.10 ±0.21
MAU-1	**90.67** ±0.16	**92.98** ±0.19	**93.35** ±0.18	**93.65** ±0.17	**93.78** ±0.14	**95.51** ±0.10	93.67 ±0.18	**94.40** ±0.17	**94.65** ±0.14
MAU-2	**90.59** ±0.15	**92.91** ±0.20	**93.40** ±0.19	**93.55** ±0.16	**93.70** ±0.16	**95.45** ±0.12	93.65 ±0.17	**94.45** ±0.19	**94.75** ±0.16
MAU-3	**90.60** ±0.17	**92.90** ±0.20	**93.45** ±0.17	**93.60** ±0.18	**93.70** ±0.15	**95.55** ±0.11	93.61 ±0.18	**94.47** ±0.16	**94.69** ±0.16

Table 6.3: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

Activation Function	ResNext	Squeezenet	AlexNet	LeNet	Xception	SeNet 18	SeNet 34	SeNet 50
ReLU	92.90 ±0.20	90.11 ±0.22	83.90 ±0.21	75.35 ±0.23	90.52 ±0.22	94.10 ±0.22	94.20 ±0.20	94.20 ±0.20
Leaky ReLU	93.12 ±0.22	90.27 ±0.24	83.89 ±0.22	75.60 ±0.22	90.63 ±0.24	94.17 ±0.22	94.27 ±0.18	94.26 ±0.22
PReLU	93.26 ±0.24	90.42 ±0.24	83.81 ±0.20	75.65 ±0.22	90.55 ±0.22	94.20 ±0.25	94.52 ±0.20	94.41 ±0.22
ReLU6	93.29 ±0.20	90.41 ±0.22	84.19 ±0.19	75.82 ±0.22	90.60 ±0.22	94.18 ±0.21	94.29 ±0.20	94.27 ±0.21
ELU	93.20 ±0.20	90.43 ±0.22	83.87 ±0.21	75.59 ±0.22	90.90 ±0.23	94.40 ±0.22	94.35 ±0.18	94.20 ±0.21
Softplus	92.70 ±0.26	90.10 ±0.28	83.59 ±0.24	75.40 ±0.27	90.52 ±0.26	94.17 ±0.26	94.21 ±0.22	93.84 ±0.23
PAU	93.75 ±0.20	90.80 ±0.21	85.20 ±0.17	76.77 ±0.20	90.91 ±0.21	94.72 ±0.24	94.97 ±0.21	94.95 ±0.22
Swish	93.70 ±0.21	90.99 ±0.21	85.15 ±0.22	**77.10** ±0.22	90.80 ±0.20	94.78 ±0.20	94.82 ±0.23	94.90 ±0.24
GELU	93.80 ±0.23	90.72 ±0.22	85.14 ±0.20	76.67 ±0.22	90.88 ±0.22	94.79 ±0.22	94.90 ±0.22	95.17 ±0.22
MAU-1	**94.55** ±0.21	**91.60** ±0.18	**86.12** ±0.18	77.05 ±0.21	**91.80** ±0.19	**95.50** ±0.20	**95.39** ±0.18	**95.75** ±0.21
MAU-2	**94.67** ±0.22	**91.77** ±0.20	**86.17** ±0.21	77.01 ±0.20	**91.78** ±0.20	**95.59** ±0.18	**95.47** ±0.18	**95.65** ±0.22
MAU-3	**94.60** ±0.21	**91.65** ±0.20	**86.22** ±0.19	77.02 ±0.22	**91.64** ±0.23	**95.60** ±0.18	**95.36** ±0.20	**95.61** ±0.21

Table 6.4: Experimental Results for MAU-1, MAU-2, MAU-3 and Baseline Activations in CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

Activation Function	Inception V3	WideRes Net 28-10	DenseNet 121	Effitient Net B0	VGG16	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50
ReLU	93.97 ±0.20	94.81 ±0.20	94.35 ±0.17	94.50 ±0.18	93.19 ±0.24	93.37 ±0.20	93.99 ±0.17	94.07 ±0.18
Leaky ReLU	93.87 ±0.21	94.90 ±0.21	94.50 ±0.16	94.44 ±0.20	93.15 ±0.22	93.49 ±0.20	93.85 ±0.17	94.21 ±0.20
PReLU	93.85 ±0.20	94.90 ±0.23	94.60 ±0.20	94.75 ±0.21	93.25 ±0.26	93.35 ±0.23	91.29 ±0.19	94.49 ±0.22
ReLU6	93.90 ±0.20	95.22 ±0.20	94.79 ±0.22	94.41 ±0.22	93.29 ±0.20	93.51 ±0.20	93.89 ±0.18	94.52 ±0.18
ELU	93.78 ±0.20	95.21 ±0.20	94.77 ±0.20	94.45 ±0.22	93.20 ±0.21	93.70 ±0.24	93.79 ±0.20	94.55 ±0.18
Softplus	93.40 ±0.23	94.65 ±0.24	94.40 ±0.21	94.65 ±0.22	93.20 ±0.26	93.33 ±0.26	91.19 ±0.24	94.31 ±0.25
PAU	94.17 ±0.19	94.45 ±0.20	94.90 ±0.20	94.85 ±0.22	93.31 ±0.22	94.52 ±0.20	94.50 ±0.22	94.52 ±0.20
Swish	94.00 ±0.20	94.69 ±0.23	94.77 ±0.22	94.67 ±0.24	93.64 ±0.22	**94.54** ±0.20	94.60 ±0.22	94.60 ±0.21
GELU	94.01 ±0.20	94.57 ±0.24	94.81 ±0.23	94.69 ±0.21	93.60 ±0.22	94.50 ±0.23	94.40 ±0.22	94.60 ±0.22
MAU-1	**94.55** ±0.12	**95.65** ±0.13	**95.30** ±0.12	**95.46** ±0.15	**93.99** ±0.20	94.47 ±0.16	**95.17** ±0.15	**95.04** ±0.15
MAU-2	**94.49** ±0.11	**95.55** ±0.14	**95.31** ±0.14	**95.40** ±0.16	**93.97** ±0.19	94.42 ±0.17	**95.18** ±0.16	**95.15** ±0.16
MAU-3	**94.59** ±0.10	**95.61** ±0.15	**95.39** ±0.12	**95.49** ±0.16	**93.95** ±0.18	94.50 ±0.18	**95.29** ±0.14	**95.10** ±0.14

Table 6.5: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR10 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

Activation Function	ResNext	Squeezenet	AlexNet	LeNet	Xception	SeNet 18	SeNet 34	SeNet 50
ReLU	74.18 ±0.22	65.83 ±0.23	54.50 ±0.28	45.29 ±0.27	70.79 ±0.22	74.30 ±0.21	75.10 ±0.22	75.80 ±0.21
Leaky ReLU	74.40 ±0.27	66.30 ±0.24	54.81 ±0.25	45.21 ±0.28	71.30 ±0.26	74.59 ±0.25	75.31 ±0.23	76.29 ±0.23
PReLU	74.52 ±0.27	66.49 ±0.24	55.21 ±0.23	45.30 ±0.26	71.50 ±0.26	74.50 ±0.22	75.21 ±0.20	76.55 ±0.22
ReLU6	74.55 ±0.24	66.16 ±0.24	55.64 ±0.20	45.26 ±0.28	71.50 ±0.25	74.41 ±0.23	75.41 ±0.24	76.80 ±0.22
ELU	74.85 ±0.23	66.45 ±0.22	56.61 ±0.23	45.63 ±0.26	71.89 ±0.26	74.60 ±0.24	75.37 ±0.23	76.89 ±0.24
Softplus	73.80 ±0.26	66.30 ±0.28	54.47 ±0.28	45.69 ±0.32	70.65 ±0.26	74.20 ±0.23	74.77 ±0.25	75.85 ±0.24
PAU	75.79 ±0.24	66.91 ±0.22	58.07 ±0.26	46.77 ±0.27	73.21 ±0.23	74.50 ±0.21	75.27 ±0.22	77.54 ±0.17
Swish	75.49 ±0.24	66.44 ±0.23	57.59 ±0.27	46.61 ±0.25	72.61 ±0.22	74.50 ±0.21	75.60 ±0.22	77.21 ±0.17
GELU	75.59 ±0.21	66.56 ±0.20	57.77 ±0.29	46.49 ±0.29	72.91 ±0.24	74.50 ±0.24	75.59 ±0.20	77.16 ±0.20
MAU-1	**77.10** ±0.22	**68.17** ±0.20	**61.10** ±0.24	**47.24** ±0.25	**74.23** ±0.24	**75.89** ±0.21	**76.45** ±0.19	**78.80** ±0.20
MAU-2	**76.95** ±0.21	**68.20** ±0.21	**61.17** ±0.26	**47.15** ±0.24	**74.20** ±0.23	**75.98** ±0.19	**76.50** ±0.19	**78.89** ±0.20
MAU-3	**77.01** ±0.22	**68.26** ±0.20	**61.01** ±0.25	**47.55** ±0.24	**74.26** ±0.23	**75.88** ±0.21	**76.52** ±0.20	**78.81** ±0.20

Table 6.6: Experimental Results for MAU-1, MAU-2, MAU-3 and Baseline Activations in CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

Activation Function	SF V2 0.5x	SF V2 1.0x	SF V2 1.5x	SF V2 2.0x	MobileNet V1	MobileNet V2	ResNet 18	ResNet 34	ResNet 50
ReLU	61.85 ±0.26	64.20 ±0.29	66.65 ±0.28	66.85 ±0.25	70.94 ±0.26	74.02 ±0.26	73.12 ±0.24	73.25 ±0.25	73.85 ±0.23
Leaky ReLU	61.81 ±0.28	64.30 ±0.30	66.50 ±0.26	67.40 ±0.26	71.06 ±0.25	74.15 ±0.24	73.20 ±0.23	73.40 ±0.23	74.26 ±0.21
PReLU	62.23 ±0.26	64.20 ±0.29	66.98 ±0.29	67.79 ±0.24	71.21 ±0.26	74.20 ±0.26	73.45 ±0.24	73.62 ±0.23	74.48 ±0.25
ReLU6	62.20 ±0.27	64.34 ±0.24	66.84 ±0.26	67.67 ±0.26	71.40 ±0.23	74.42 ±0.23	73.12 ±0.24	73.12 ±0.23	74.51 ±0.22
ELU	62.19 ±0.28	64.60 ±0.26	66.55 ±0.26	67.72 ±0.22	71.27 ±0.26	74.46 ±0.27	73.32 ±0.23	73.59 ±0.24	74.41 ±0.25
Softplus	61.80 ±0.32	64.50 ±0.31	66.63 ±0.31	67.31 ±0.30	70.90 ±0.27	74.15 ±0.26	73.30 ±0.24	73.29 ±0.27	74.22 ±0.22
PAU	63.35 ±0.26	66.48 ±0.26	69.25 ±0.26	70.20 ±0.25	71.41 ±0.28	74.65 ±0.24	74.50 ±0.20	73.52 ±0.22	75.61 ±0.21
Swish	63.13 ±0.25	66.26 ±0.24	69.15 ±0.27	70.71 ±0.24	71.50 ±0.24	74.57 ±0.25	**74.64** ±0.23	74.20 ±0.21	75.52 ±0.22
GELU	63.40 ±0.26	66.20 ±0.24	69.30 ±0.28	70.81 ±0.24	71.31 ±0.26	74.60 ±0.24	74.60 ±0.24	73.99 ±0.23	75.41 ±0.22
MAU-1	**64.56** ±0.24	**68.64** ±0.23	**72.10** ±0.26	**72.60** ±0.23	**72.10** ±0.22	**75.89** ±0.20	74.48 ±0.22	**74.69** ±0.24	**76.56** ±0.21
MAU-2	**64.50** ±0.23	**68.56** ±0.24	**72.12** ±0.24	**72.51** ±0.24	**72.21** ±0.23	**75.97** ±0.22	74.45 ±0.24	**74.75** ±0.25	**76.60** ±0.22
MAU-3	**64.60** ±0.25	**68.50** ±0.22	**72.11** ±0.25	**72.65** ±0.25	**72.01** ±0.23	**75.80** ±0.21	74.49 ±0.23	**74.59** ±0.22	**76.51** ±0.20

Table 6.7: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

Activation Function	Inception V3	WideRes Net 28-10	DenseNet 121	Effitient Net B0	VGG16	PreAct ResNet 18	PreAct ResNet 34	PreAct ResNet 50
ReLU	74.12 ±0.26	75.95 ±0.24	75.80 ±0.28	76.20 ±0.23	71.16 ±0.31	72.60 ±0.26	72.99 ±0.23	73.32 ±0.25
Leaky ReLU	74.30 ±0.28	75.85 ±0.26	75.81 ±0.25	76.37 ±0.24	71.15 ±0.28	72.75 ±0.21	72.91 ±0.27	73.50 ±0.24
PReLU	74.30 ±0.29	75.99 ±0.26	76.15 ±0.25	76.54 ±0.27	71.20 ±0.30	72.91 ±0.25	73.51 ±0.25	73.81 ±0.25
ReLU6	74.31 ±0.25	75.81 ±0.25	75.81 ±0.22	76.26 ±0.24	71.19 ±0.27	72.55 ±0.25	73.101 ±0.24	73.60 ±0.23
ELU	74.40 ±0.24	76.25 ±0.26	75.75 ±0.25	76.24 ±0.27	71.15 ±0.28	72.10 ±0.27	73.17 ±0.28	74.18 ±0.25
Softplus	74.09 ±0.30	75.45 ±0.27	75.86 ±0.26	75.70 ±0.28	71.21 ±0.30	71.84 ±0.30	73.10 ±0.28	74.32 ±0.32
PAU	75.19 ±0.21	75.87 ±0.24	76.23 ±0.23	76.60 ±0.25	71.59 ±0.24	74.12 ±0.25	73.91 ±0.24	75.48 ±0.24
Swish	74.70 ±0.24	75.85 ±0.23	75.87 ±0.24	76.45 ±0.23	71.97 ±0.24	74.15 ±0.24	74.41 ±0.26	75.65 ±0.23
GELU	74.70 ±0.24	76.30 ±0.23	76.35 ±0.24	76.91 ±0.23	71.60 ±0.27	**74.22** ±0.23	74.32 ±0.23	75.47 ±0.24
MAU-1	**76.12** ±0.20	**77.54** ±0.20	**77.19** ±0.24	**78.15** ±0.24	**73.12** ±0.26	74.10 ±0.23	**75.52** ±0.25	**76.78** ±0.21
MAU-2	**76.19** ±0.19	**77.47** ±0.22	**77.16** ±0.23	**78.20** ±0.24	**73.01** ±0.24	74.07 ±0.24	**75.50** ±0.25	**76.70** ±0.22
MAU-3	**76.01** ±0.21	**77.41** ±0.20	**77.06** ±0.23	**78.23** ±0.26	**73.10** ±0.25	74.01 ±0.22	**75.61** ±0.23	**76.65** ±0.20

Table 6.8: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation and Other Baseline Activations on CIFAR100 Dataset for Image Classification on Different Models. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

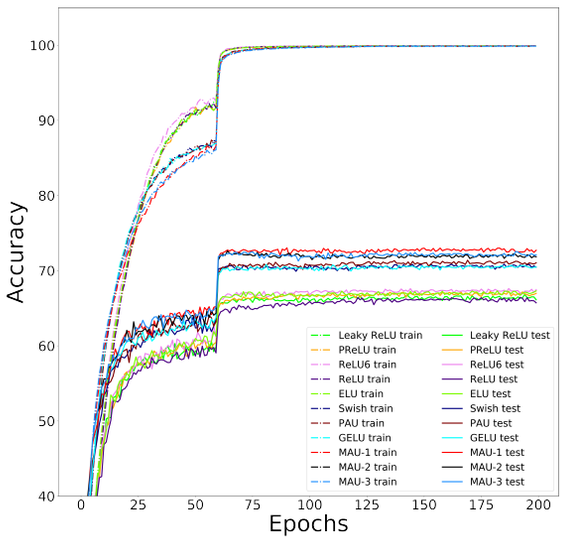

Figure 6.2: Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations and the proposed activations.

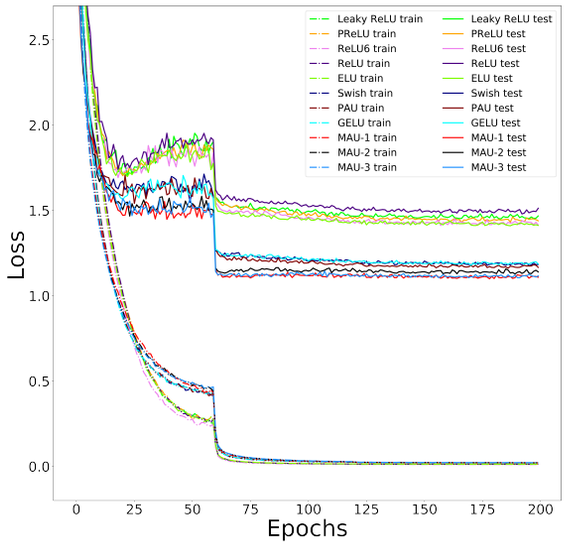

Figure 6.3: Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with Shufflenet V2 (2.0x) network for different baseline activations and the proposed activations.

Activation Function	ShuffleNet V2 (2.0x)	ResNet 50	ResNet 18
ReLU	69.20 ± 0.23	75.15 ± 0.24	73.97 ± 0.23
Leaky ReLU	69.15 ± 0.22	75.12 ± 0.23	73.90 ± 0.25
PReLU	69.30 ± 0.24	75.20 ± 0.24	74.20 ± 0.24
ReLU6	69.45 ± 0.24	75.30 ± 0.23	74.20 ± 0.22
ELU	69.45 ± 0.23	75.42 ± 0.26	74.15 ± 0.26
Softplus	68.85 ± 0.30	74.40 ± 0.25	73.81 ± 0.27
Swish	72.87 ± 0.20	76.55 ± 0.25	74.50 ± 0.24
GELU	72.90 ± 0.22	76.67 ± 0.24	74.65 ± 0.22
PAU	73.15 ± 0.23	76.80 ± 0.23	74.70 ± 0.26
MAU-1	**74.45** ± 0.20	**77.99** ± 0.20	**75.89** ± 0.18
MAU-2	**74.54** ± 0.20	**78.01** ± 0.21	**75.78** ± 0.22
MAU-3	**74.46** ± 0.18	**77.89** ± 0.20	**75.74** ± 0.21

Table 6.9: Top-1 Test Accuracy Reported with Mixup Augmentation Method on CIFAR100 Dataset for the Mean of 20 Different Runs. I report mean±std in the Table.

Tiny Imagenet:

Tiny Imagenet dataset (Le and Yang (2015)) is a similar kind of image classification database like the ImageNet Large Scale Visual Recognition Challenge(ILSVRC) (Deng *et al.* (2009)). This section presents results on the Tiny ImageNet dataset. Tiny Imagenet has a total of 200 image classes. I report the results for top-1 accuracy on table 6.10 for a mean of 20 different runs. I consider WideResNet 28-10 (WRN 28-10)

(Zagoruyko and Komodakis (2016)) and ResNet 18 (He *et al.* (2015*a*)) models to report results. I consider SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)), He Normal initializer (He *et al.* (2015*b*)), a batch size of 64, 0.2 dropout rate (Srivastava *et al.* (2014)), initial learning rate(lr rate) 0.1, and lr rate is reduced by a factor of 10 after every 50 epochs up-to 300 epochs. To improve performance, I consider standard data augmentation techniques like width shift, height shift, shearing, rotation, zoom, horizontal flip, and fill mode. It is quite clear from the table that the proposed function performs better than the baseline functions, and top-1 accuracy is stable (mean±std) for all models. I got a good improvement for MAU-1, MAU-2, and MAU-3 over ReLU.

Activation Function	WideResNet 28-10	ResNet 18
ReLU	62.65 ± 0.44	58.20 ± 0.40
Leaky ReLU	62.87 ± 0.47	58.51 ± 0.42
PReLU	62.52 ± 0.48	58.57 ± 0.45
ReLU6	62.41 ± 0.48	58.62 ± 0.40
ELU	62.67 ± 0.52	58.51 ± 0.40
Softplus	61.51 ± 0.62	58.24 ± 0.49
PAU	63.50 ± 0.46	59.58 ± 0.42
Swish	63.57 ± 0.46	59.19 ± 0.43
GELU	63.20 ± 0.47	59.36 ± 0.40
MAU-1	**64.50** ± 0.41	**60.61** ± 0.42
MAU-2	**64.57** ± 0.43	**60.65** ± 0.41
MAU-3	**64.01** ± 0.46	**60.19** ± 0.43

Table 6.10: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activation's in Tiny ImageNet Dataset for Image Classification Problem. I Report top-1 Test Accuracy (in %) for the Mean of 20 Different Runs. I report mean±std in the Table.

ImageNet-1k:

ImageNet-1k has more than 1.2 million training images with 1000 classes. Results are reported with ResNet-50 He *et al.* (2015*a*) and ShuffleNet V2 Ma *et al.* (2018) models in Table 6.11 in this dataset. I consider SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)), 0.9 momentum, $5e^{-4}$ weight decay, and a batch size of 256. I consider a linear decay learning rate scheduler from 0.1 and trained upto 600k

iterations. Experiments are conducted on four NVIDIA V100 GPUs with 32GB RAM each.

Activation Function	ShuffleNet V2 (1.0x)	ResNet-50
ReLU	69.20	75.25
Leaky ReLU	69.34	75.49
PReLU	69.23	75.31
ReLU6	69.47	75.55
ELU	69.49	75.66
Softplus	69.20	75.49
Swish	70.52	76.55
GELU	70.43	76.21
PAU	70.54	76.29
MAU-1	**71.20**	**77.32**
MAU-2	**71.32**	**77.17**
MAU-3	**71.01**	**77.16**

Table 6.11: top-1 Accuracy Reported on ImageNet-1k Dataset.

6.5.2 Object Detection

In this section, I report results on challenging Pascal VOC dataset (Everingham *et al.* (2010)) on Single Shot MultiBox Detector(SSD) 300 (Liu *et al.* (2016)). I consider VGG-16 (with batch-normalization) (Simonyan and Zisserman (2015)) as the backbone network. W do not use any pre-trained weights for our experiments in the network. I consider SGD optimizer (Robbins and Monro (1951), Kiefer and Wolfowitz (1952)) with 0.9 momentum, $5e^{-4}$ weight decay, 0.001 learning rate, a batch size of 8, and trained up to 120000 iterations. The results are reported for the mean average precision (mAP) in Table 6.12 for the mean of 12 different runs.

6.5.3 Semantic Segmentation

Semantic segmentation is a computer vision problem that narrates the procedure of associating each pixel of an image with a class label. I present our experimental results in this section on the popular Cityscapes dataset (Cordts *et al.* (2016)). The U-net model (Ronneberger *et al.* (2015)) is considered as the segmentation framework and is trained

Activation Function	mAP
ReLU	77.2±0.16
Leaky ReLU	77.2±0.18
PReLU	77.2±0.19
ReLU6	77.1±0.18
ELU	75.1±0.20
Softplus	74.2±0.23
PAU	77.4±0.15
Swish	77.3±0.13
GELU	77.3±0.12
MAU-1	**78.3**±0.10
MAU-2	**78.3**±0.11
MAU-3	**78.0**±0.12

Table 6.12: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations on Pascal VOC Dataset for Object Detection Problem with SSD300. I Report mAP for the Mean of 12 Different Runs. I report mean±std in the Table.

up-to 250 epochs, with adam optimizer (Kingma and Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)). I report the mean of 12 different runs for Pixel Accuracy and the mean Intersection-Over-Union (mIOU) on test data in table 6.13.

6.5.4 Machine Translation

In this section, I report results on WMT 2014 English→German dataset. To evaluate network performance, I consider the newstest2014 dataset using the BLEU score metric. I consider an attention-based 8-head transformer network (Vaswani *et al.* (2017)) and I consider Adam optimizer (Kingma and Ba (2015)), 0.1 dropout rate (Srivastava *et al.* (2014)). The network is trained up to 100000 steps. I try to keep the other hyper-parameters similar as mentioned in the original paper (Vaswani *et al.* (2017)). I report the mean of 12 different runs on Table 6.14 on the test dataset(newstest2014).

Activation Function	Pixel Accuracy	mIOU
ReLU	79.54±0.45	69.52±0.29
PReLU	78.96±0.42	68.99±0.35
ReLU6	79.81±0.39	69.98±0.40
Leaky ReLU	79.62±0.42	69.95±0.43
ELU	79.49±0.50	68.39±0.41
Softplus	78.51±0.53	68.28±0.55
PAU	79.69±0.50	69.35±0.32
Swish	80.12±0.46	70.01±0.30
GELU	80.01±0.41	69.61±0.35
MAU-1	**81.85**±0.38	**71.58**±0.30
MAU-2	**81.65**±0.40	**71.52**±0.32
MAU-3	**81.29**±0.41	**71.01**±0.32

Table 6.13: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations for Semantic Segmentation Problem in CityScapes Dataset with U-NET Model. I Report Pixel Accuracy and mIOU for the Mean of 12 Different Runs. mean±std is Reported in the Table.

Activation Function	BLEU Score on the newstest2014 dataset
ReLU	26.2±0.16
Leaky ReLU	26.3±0.16
PReLU	26.2±0.23
ReLU6	26.1±0.16
ELU	25.1±0.18
Softplus	23.6±0.18
PAU	26.3±0.13
Swish	26.4±0.11
GELU	26.5±0.17
MAU-1	**26.8**±0.12
MAU-2	**26.7**±0.12
MAU-3	**26.6**±0.12

Table 6.14: A Detailed Comparison between MAU-1, MAU-2, and MAU-3 Activation Functions and Other Baseline Activations in WMT-2014 Dataset for Machine Translation Problem on Transformer Model. I Report BLEU Score for the Mean of 12 Different Runs. mean±std is Reported in the Table.

Baselines	ReLU	Leaky ReLU	PReLU	ReLU6	ELU	Softplus	PAU	Swish	GELU
MAU-1 $>$ Baseline	66	66	66	66	66	66	62	60	61
MAU-1 $=$ Baseline	0	0	0	0	0	0	0	0	0
MAU-1 $<$ Baseline	0	0	0	0	0	0	4	6	5
MAU-2 $>$ Baseline	66	66	66	66	66	66	62	60	61
MAU-2 $=$ Baseline	0	0	0	0	0	0	0	0	0
MAU-2 $<$ Baseline	0	0	0	0	0	0	4	6	5
MAU-3 $>$ Baseline	66	66	66	66	66	66	62	60	61
MAU-3 $=$ Baseline	0	0	0	0	0	0	0	0	0
MAU-3 $<$ Baseline	0	0	0	0	0	0	4	6	5

Table 6.15: Baseline Table for MAU-1, MAU-2, & MAU-3. In the Table, I Report the Total Number of Cases in Which the proposed functions Underperforms, Equal, or Outperforms When compared with the Baseline Activation Functions

6.6 Baseline Table

In this section, I present the baseline table for MAU-1, MAU-2, & MAU-3 and the other baseline functions, which shows that MAU-1, MAU-2, & MAU-3 beat the baseline activation functions in most cases and underperform in marginal cases. A detailed comparison with MAU-1, MAU-2, & MAU-3 and the baseline activation functions based on all the experiments in previous sections has been reported in Table 6.15. From the baseline table, it is clear that the proposed functions perform well in most of the cases compared to the baseline activation function.

6.7 Computational Time Comparison

This section presents the computational time comparison for the baseline activation functions and MAU-1, MAU-2, ad MAU-3 for the mean of 100 runs. The results are reported for both forward and backward pass on a 32×32 RGB image in PreActResNet-18 He *et al.* (2016) model in Table 6.16. An NVIDIA Tesla V100 GPU with 32GB ram is considered to run the experiments.

Note that, here is a small trade-off between running time and model performance for the proposed activation functions and the activation functions like ReLU, Leaky ReLU, ReLU6. It Is quite clear from the Table 6.16 and the experiment section. Also, note that the time is almost similar to other smooth activations like Swish, Mish or GELU &

better than PAU.

Activation Function	Forward Pass	Backward Pass
ReLU	5.50 ± 0.41 μs	5.72 ± 1.63 μs
Swish	8.15 ± 1.54 μs	10.40 ± 2.25 μs
Leaky ReLU	5.56 ± 0.50 μs	6.04 ± 0.71 μs
PReLU	6.21 ± 0.91 μs	6.31 ± 1.31 μs
ReLU6	5.98 ± 0.75 μs	6.01 ± 0.71 μs
ELU	6.18 ± 0.52 μs	5.99 ± 0.95 μs
Softplus	6.19 ± 0.49 μs	6.10 ± 0.51 μs
GELU	8.59 ± 1.48 μs	9.19 ± 1.69 μs
PAU	19.16 ± 2.65 μs	32.71 ± 3.70 μs
MAU-1	7.81 ± 1.51 μs	10.62 ± 1.53 μs
MAU-2	7.69 ± 1.83 μs	10.95 ± 1.70 μs
MAU-3	7.41 ± 1.45 μs	10.53 ± 1.70 μs

Table 6.16: Runtime comparison for the forward and backward passes for MAU-1, MAU-2, & MAU-3 and baseline activation functions for a $32\times$ 32 RGB image in PreActResNet-18 model.

6.8 Conclusion

In this chapter, three smooth activation functions have been presented. The maximum function is not smooth. A special form of the maximum function is derived and then approximated the form by known smooth functions. I call these three functions as Maximum Activation Unit (MAU). The proposed functions are smooth and can approximate ReLU or its different variants quite well. By experimental evaluation, in a large number of datasets on different deep learning problems, it has been shown that the proposed functions perform better than the known widely used activations like ReLU, Leaky ReLU or Swish in most cases which shows that replacing the hand-crafted activation functions by MAU-1, MAU-2, and MAU-3 can be beneficial in deep networks.

CHAPTER 7

Orthogonal-Padé Activation Unit [1]

7.1 Introduction

Deep networks are constructed with multiple hidden layers and neurons. Non-linearity is introduced in the network via activation function in each neuron. ReLU (Nair and Hinton (2010)) is proposed by Nair and Hinton and is the favourite activation in the deep learning community due to its simplicity. Though ReLU has a drawback called dying ReLU, and in this case, up to 50% neurons can be dead due to the vanishing gradient problem, i.e. there are numerous neurons which have no impact on the network performance. To overcome this problem, later Leaky Relu (Maas *et al.* (2013*a*)), Parametric ReLU (He *et al.* (2015*b*)), ELU (Clevert *et al.* (2016)), and Softplus (Zheng *et al.* (2015)) were proposed, and they have improved the network performance through its still an open problem for researchers to find the best activation function. Recently Swish (Ramachandran *et al.* (2017)) was found by a group of researchers from Google brain, and they used an automated searching technique. Swish has shown some improvement in accuracy over ReLU. GELU (Hendrycks and Gimpel (2020)), and Mish (Misra (2020)) are a few other candidates proposed recently which can replace ReLU and Swish. Recently, there has been an increasing interest in trainable activation functions. Trainable activation functions have learnable hyperparameter(s), updated during training via backpropagation algorithm (LeCun *et al.* (1989)). In this chapter, we have proposed Orthogonal-Padé activation functions. Orthogonal-Padé functions can approximate most of the continuous functions.

7.2 Related works and motivation

Activation functions are a crucial component of a neural network. It introduces the non-linearity in a network. ReLU (Nair and Hinton (2010)) is a popular non-linear acti-

[1] This chapter is a slightly modified version of the paper in Arxiv Biswas *et al.* (2021*a*).

vation function though it has some drawbacks. Leaky ReLU (Maas *et al.* (2013*a*)) and Parametric ReLU (He *et al.* (2015*b*)) are proposed later to overcome the drawback of ReLU. Swish (Ramachandran *et al.* (2017)) is a recently proposed activation proposed by the Google Brain team. All these functions are handcrafted though there is another way of proposing an activation function using the approximation method, which is not explored much. Padé Activation Unit (Molina *et al.* (2020) is proposed recently by approximation method using rational polynomial approximation method. Experimentally PAU shows that the proposed activation function improves performance in image classification problems.

7.3 Research contribution

In this chapter, we propose activation functions using the rational polynomial approximation technique using standard orthogonal polynomials. We replace the standard polynomial basis in Padé approximation by orthogonal basis and approximate the Leaky ReLU function. Now the approximating function can be used as an activation function. We have conducted extensive experiments to find the effects of various neural network's training dynamics and performance. Our experimental evaluation shows that our proposed activation functions are comparatively more effective than ReLU, Swish, GELU, PAU etc., across different deep learning tasks. We summarise the chapter as follows:

1. We have proposed activation functions using rational orthogonal polynomial approximation technique. We approximate the Leaky ReLU function and find the coefficients of the polynomials for the proposed functions.
2. We show that the proposed functions outperform widely used activation functions across four different deep learning tasks.

7.4 Padé activation Unit (PAU) and Orthogonal-PAU

Let $\{\mathcal{P}_n(x)\}_{n=0}^{\infty}$ be a sequence of polynomials in x with degree$\mathcal{P}_n(x) = n$ for each n. For a positive, continuous function $w(x)$ on the interval (a, b) with $a < b$, define an inner product on $\mathcal{P}_n(x)$ as

$$< P, Q >_w = \int_a^b w(x)P(x)Q(x)\, dx. \tag{7.1}$$

A finite set of polynomials $\{P_1(x), P_2(x), \cdots, P_k(x)\}$ is said to be orthogonal with respect to the weight function $w(x)$ on the interval (a, b) with $a < b$ if

$$< P_i, P_j >_w = 0 \quad \text{if } i \neq j. \tag{7.2}$$

A basis for $\mathcal{P}_n(x)$ is a set of n polynomials whose span is whole of $\mathcal{P}_n(x)$. An orthogonal basis is a basis that is also an orthogonal set. A more detailed information about orthogonal polynomials, see (Weisstein).

A standard basis for $\mathcal{P}_n(x)$ is $\{1, x, x^2, \cdots, x^n\}$. But the standard basis is not orthogonal with respect to the inner product defined in (7.1). In many applications, working with an orthogonal basis simplifies expressions and reduce calculations. There are several well known orthogonal basis for the space of polynomials. Table 7.1 enlists some of these polynomial bases. Note that some of them are given by recurrence relations and others by direct expressions.

Polynomial	**Recurrence Relation/Expression**
Chebyshev polynomial of the first kind (CP-1)	$r_0(x) = 1, r_1(x) = x, r_{n+1}(x) = 2xr_n(x) - r_{n-1}(x)$
Chebyshev polynomial of the Second kind (CP-2)	$r_0(x) = 1, r_1(x) = 2x, r_{n+1}(x) = 2xr_n(x) - r_{n-1}(x)$
Laguerre polynomials (LAU)	$r_0(x) = 1, r_1(x) = 1 - x, r_{n+1}(x) = \frac{(2n+1-x)r_n(x) - nr_{n-1}(x)}{n+1}$
Legendre polynomials (LEG)	$r_n(x) = \sum_{k=0}^{[n/2]} (-1)^k \frac{(2n-2k)!}{2^n k!(n-2k)!(n-k)} x^{n-2k}$
Probabilist's Hermite polynomials (HP-1)	$r_n(x) = (-1)^n e^{\frac{x^2}{2}} \frac{d^n}{dx^n} e^{-\frac{x^2}{2}}$
Physicist's Hermite polynomials (HP-2)	$r_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$

Table 7.1: Some well-known Orthogonal Polynomial Basis.

7.4.1 Padé activation Unit (PAU)

The Padé approximation of $f(x)$ by a rational function $F_1(x)$ is defined as

$$F_1(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} a_i x^i}{1 + \sum_{j=1}^{l} b_j x^j} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k}{1 + b_1 x + b_2 x^2 + \cdots + b_l x^l} \tag{7.3}$$

where $P(x)$ and $Q(x)$ are polynomials of degree k and l respectively and they have no common factor. PAU (Molina *et al.* (2020)) is a learnable activation function of the form given in (7.3) where the polynomial coefficients $a_i, b_j, 0 \leq i \leq k, 1 \leq j \leq l$

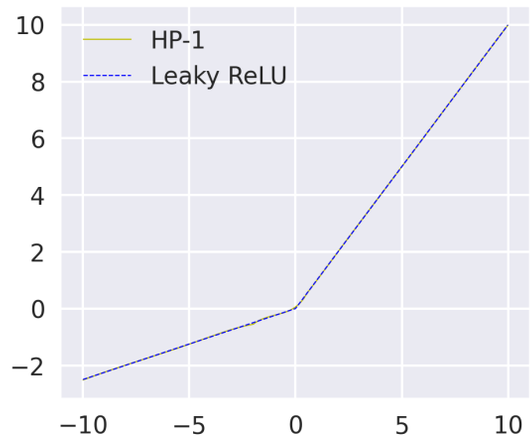

Figure 7.1: Appximation of Leaky ReLU ($\alpha = 0.25$) by HP-1 function.

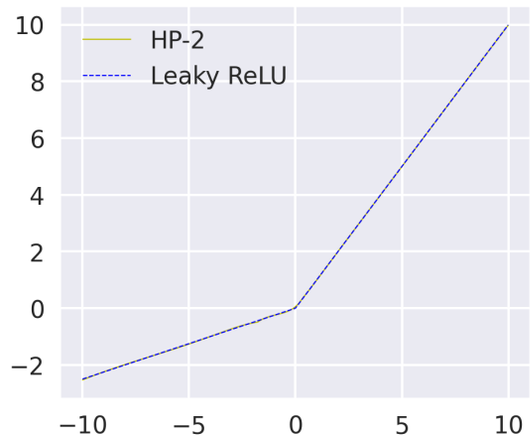

Figure 7.2: Appximation of Leaky ReLU ($\alpha = 0.25$) by HP-2 function.

are learnable parameters and updated during back-propagation. To remove the pole of $F_1(x)$ coming from zeros of $Q(x)$, authors in (Molina *et al.* (2020)) proposed safe PAU. Safe PAU is defined as

$$F_2(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} a_i x^i}{1 + |\sum_{j=1}^{l} b_j x^j|} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k}{1 + |b_1 x + b_2 x^2 + \cdots + b_l x^l|} \tag{7.4}$$

Introducing the absolute value in the denominator ensures that the denominator will not vanish. In fact, one can take absolute value inside the sum and define

$$F_3(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} a_i x^i}{1 + \sum_{j=1}^{l} |b_j||x^j|} = \frac{a_0 + a_1 x + a_2 x^2 + \cdots + a_k x^k}{1 + |b_1||x| + |b_2||x^2| + \cdots + |b_l||x^l|} \tag{7.5}$$

We will show that in many tasks activation functions defined by F_3 provide better results than safe PAU defined in F_2.

7.4.2 Orthogonal-Padé activation Unit (OPAU)

The orthogonal-Padé approximation of $g(x)$ by a rational function $G(x)$ is defined as

$$G(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} c_i f_i(x)}{1 + \sum_{j=1}^{l} dj f_j(x)} = \frac{c_0 + c_1 f_1(x) + c_2 f_2(x) + \cdots + c_k f_k(x)}{1 + d_1 f_1(x) + d_2 f_2(x) + \cdots + d_l f_l(x)} \tag{7.6}$$

where $f_t(x)$ belongs to a set of orthogonal polynomials (see Weisstein). As in the case of PAU, the learnable activation function, OPAU, is defined by (7.6) where $c_i, d_j, 0 \leq$

$i \leq k, 1 \leq j \leq l$ are learnable parameters. The parameters are initialized by taking approximation of the form (7.6) of a well-known activation function like ReLU, Leaky ReLU etc., see (Molina *et al.* (2020)). To remove poles of $G(x)$, we propose safe OPAU as follows:

$$G(x) = \frac{P(x)}{Q(x)} = \frac{\sum_{i=0}^{k} c_i f_i(x)}{1 + \sum_{j=1}^{l} |dj||f_j(x)|} = \frac{c_0 + c_1 f_1(x) + c_2 f_2(x) + \cdots + c_k f_k(x)}{1 + |d_1||f_1(x)| + |d_2||f_2(x)| + \cdots + |d_l||f_l(x)|} \tag{7.7}$$

We consider six orthogonal polynomial bases - Chebyshev (two types), Hermite (two types), Laguerre, and Legendre polynomial bases for this work. Details about these polynomial bases are given in Table 7.1.

7.4.3 Learning activation parameters via back-propagation

Weights and biases in neural network models are updated via backpropagation algorithm and gradient decent. The same method is adopted to update the activation parameters. We implement the forward pass in both Pytorch (Paszke *et al.* (2019)) & Tensorflow-Keras (Chollet *et al.* (2015)) API and automatic differentiation will update the parameters. Alternatively, CUDA (Nickolls *et al.* (2008)) based implementation (see (Molina *et al.* (2020)), (Maas *et al.* (2013*a*))) can be used and the gradients of equations (7.6) for the input x and the parameters c_i's and d_j's can be computed as follows:

$$\frac{\partial G}{\partial x} = \frac{1}{Q(x)} \frac{\partial P(x)}{\partial x} - \frac{P(x)}{Q(x)^2} \frac{\partial Q(x)}{\partial x}, \quad \frac{\partial G}{\partial c_i} = \frac{f_i(x)}{Q(x)}, \quad \frac{\partial G}{\partial d_j} = -\operatorname{sgn}(d_j)|f_j(x)| \frac{P(x)}{Q(x)^2}. \tag{7.8}$$

7.5 Networks with orthogonal-Padé activations and function approximation

Orthogonal-Padé networks are similar to Padé networks (Molina *et al.* (2020)) in which a network with PAU or safe PAU is replaced with an OPAU or safe OPAU. In this article, we consider safe OPAUs' as an activation function with different orthogonal bases as given in Table 7.1. We initialize the learnable parameters (polynomial coefficients)

using the approximation of Leaky ReLU ($\alpha = 0.01$) by the functional form given in (7.7). The network parameters are optimized via the backpropagation method (LeCun *et al.* (1989)). We kept a similar design for all networks as PAU in (Molina *et al.* (2020)), for example, weight sharing and learning activation parameter per layer (Teh and Hinton (2000)). From equation (7.7), we have a total of $(k + l)$ extra parameters per layer. Therefore, if there are L layers in a network, there will be extra $L \times (k + l)$ numbers of learnable parameters in the network. To train a network, we adopt Leaky ReLU initialization ($\alpha = 0.01$) instead of the random initialization method, and results are reported in the experiments section. A plot of HP-1 and HP-2 are given in figure 7.1 and figure 7.2.

Also, note that the class of neural networks with safe OPAU activation functions is dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K.
The proof follows from the following propositions (see Molina *et al.* (2020)).

Proposition 1. (Theorem 1.1 in Kidger and Lyons, 2019 Kidger and Lyons (2020)) :- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

Proposition 2. (From Theorem 3.2 in (Kidger and Lyons, 2019 Kidger and Lyons (2020))):- Let $\rho : \mathbb{R} \to \mathbb{R}$ be any continuous function which is continuously differentiable at at least one point, with nonzero derivative at that point. Let $K \subseteq \mathbb{R}^n$ be compact. Then $NN_{n,m,n+m+2}^\rho$ is dense in $C(K; \mathbb{R}^m)$.

7.6 Appximation coefficients for different orthogonal polynomials

The coefficients for different orthogonal polynomials given in Table 7.1 are reported in Table 7.2. The coefficients are found with orthogonal polynomial basis for rational function approximation (using equation (6)) to Leaky ReLU ($\alpha = 0.01$) activation function. We have computed the orthogonal polynomials (from Table 7.1 using recurrence

relations) for $k = 5$ and $l = 4$ in equation 7.7. The least-square method is adopted to optimize the error between Leaky ReLU and the rational function with orthogonal polynomials.

PC	CP-1	CP-2	LAU	LEG	HP-1	HP-2
c_0	0.43463 381995 28298	0.26646 729134 92625	1.83604 452353 54788	0.32073 3733020 75475	1.13719 634240 21352	0.46209 15542 74137
c_1	0.75822 186996 82254	0.34803 0470194 67215	-2.9554 5059092 67266	0.71427 996686 06886	1.79794 191284 49188	0.48393 211064 20414
c_2	0.31781 494330 90529	0.16180 67408 60617	1.63873 68018 88696	0.42468 163573 28257	1.1020 7705501 87182	0.18164 108628 37883
c_3	0.05703 79742924 44685	0.030197 9928897 31528	-0.317749 758837 76296	0.023434 0936823 45926	0.32948 857204 34351	0.03037 625251 52446
c_4	0.004000 91162698 71334	0.002163 1764095 56791	-0.023982 81897 0702	0.007618 7459904 66922	0.04271 8579950 60412	0.002074 690747 081737
c_5	9.93204 2145345 177e-05	5.44252 19890802 244e-05	0.01114 23449225 87972	0.000212 0535423 305138	0.002084 0356797 464945	5.145762 05169 9321e-05
d_1	-0.42263 7203997 40756	0.167403 991429 00575	-0.58902 621993 20808	0.35334 1300183 60843	1.08464 598880 19664	0.240243 594312 60522
d_2	0.14463 24151 547079	0.08512 431596 790718	-0.09392 2337654 24439	0.214676 829578 40964	0.30850 1565523 30404	0.075156 681726 28485
d_3	-0.006010 64666153 19236	0.002646 1214606 926624	0.003915 139808 859812	0.00861 13281499 30994	-0.041635 9246952 19075	0.003128 166547 86619
d_4	0.000244 05206679 94119	0.000148 137501455 71406	0.006420 352790 087902	0.0005072 095551 410509	0.002240 515203 527783	0.000127 09353203 643316

Table 7.2: Coefficient Table for Leaky ReLU rational function approximation with orthogonal basis (using equation (6)) for network initialization. 'PC' stands for polynomial coefficients.

7.7 Experimental results with Orthogonal-Padé Activation

We first initialize the trainable parameters (polynomial coefficients) of safe OPAU activation functions by rational function approximation of Leaky ReLU ($\alpha = 0.01$) activation and then update the parameters via backpropagation algorithm via 7.8. The coefficients for Leaky ReLU ($\alpha = 0.01$) approximation are given in Table 7.2. Also, from our experiments, we notice that orthogonal basis approximate a continuous function uniquely much faster than the standard polynomial basis.

Also, note that widely used activation functions in most cases are zero centered. We impose some conditions on safe PAU and safe OPAU approximation to make the known function approximation (in our case, Leaky ReLU initialization) zero centered and check whether there is any advantage (one definite advantage is the number of parameters reduces in each layer) on model performance. To make safe Padé zero centered, we first replace $a_0 = 0$ in equation 7.4 and then we find the rest of the parameters $a_1, a_2, \cdots, a_k$ and $b_1, b_2, \cdots, b_l$ by approximation of the Leaky ReLU function. For safe OPAU, several cases arrive, and we explore all possible cases. For example, if we choose HP-1 as a basis, the safe OPAU function approximation can be zero centered if the constant term in the numerator is zero. So, we have from equation 7.7 and Table 7.1, $c_0 - c_2 + 3c_4 = 0$, which means either $c_0 = c_2 = c_4 = 0$ or one of c_0 or c_2 or c_4 is equal to zero or $c_0 = c_2 - 3c_4$ or $c_2 = c_0 + 3c_4$ or $c_4 = \frac{1}{3}(c_2 - c_0)$.
In all the above cases for HP-1, and $a_0 = 0$ for PAU, the rational function approximation to Leaky ReLU ($\alpha = 0.01$) are explored and tested on CIFAR10 (Krizhevsky (2009)) and CIFAR100 (Krizhevsky (2009)) datasets on several standard networks like PreActResNet-34 (He *et al.* (2016)), LeNet (Lecun *et al.* (1998)), AlexNet (Krizhevsky *et al.* (2012)), DenseNet-121 (Huang *et al.* (2016*a*)), and EfficientNet B0 (Tan and Le (2020)) networks. We find that in most cases, network performance in top-1 accuracy reduces by 0.2%-0.6%.

In the next subsections, we give details of our experimental setup, experimental results on different deep learning problems like image classification, object detection, semantic segmentation, and machine translation in some widely used standard datasets

and networks. We consider ReLU (Nair and Hinton (2010)), Leaky ReLU (Maas *et al.* (2013*a*)), ELU (Clevert *et al.* (2016)), Softplus (Clevert *et al.* (2016)), PReLU (He *et al.* (2015*b*)), GELU (Hendrycks and Gimpel (2020)), ReLU6 (Krizhevsky (2010)), PAU (Molina *et al.* (2020)) and Swish (Ramachandran *et al.* (2017)) as our baseline activation functions to compare performance on different networks. From all our experiments, we notice that HP-1 and HP-2 continuously outperform CP-1, CP-2, LAU, and LEG activations. Also, theoretically there are some benefits of using Hermite-Padé approximation over Padé approximation (for details, please see (Beckermann *et al.* (2011))). For all our experiments, we choose networks and hyper-parameters with ReLU activation and replace ReLU by baselines and safe OPAU activations to compare network performances.

7.7.1 Image Classification

MNIST, Fashion MNIST, and The Street View House Numbers (SVHN) Database

The MNIST (LeCun *et al.* (2010)), and Fashion MNIST (Xiao *et al.* (2017)) are popular computer vision database and both databases contains a total of 60k training and 10k testing 28×28 grey-scale spread over ten different classes. MNIST contains handwritten digits from 0 to 9, while Fashion MNIST contains fashion items. SVHN (Netzer *et al.* (2011)) consists of 32×32 RGB images of real-world house numbers of Google's street view images with a total of 73257 training images and 26032 testing images, and the images are spread over 10 different classes. The data augmentation method is not used in MNIST and Fashion MNIST, while the data augmentation method is used in SVHN database. Results on VGG-16 (Simonyan and Zisserman (2015)) (with batch-normalization) network are reported in Table 7.3. We have carried out more experiments on AlexNet (AN) (Krizhevsky *et al.* (2012)), LeNet (Lecun *et al.* (1998)), and with a custom 8-layer homogeneous convolutional neural network (CNN) architecture on MNIST, Fashion MNIST, and SVHN datasets on Table 7.4, 7.5, and 7.6 respectively. The custom network is constructed with 3×3 kernels on CNN layers and pooling layers with 2×2 kernels. Channel depths of size 128 (twice), 64 (thrice), 32 (twice), a dense layer of size 128, Max-pooling layer(thrice) are used with batch-normalization (Ioffe and Szegedy (2015)), and dropout (Srivastava *et al.* (2014)).

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.01 ± 0.10	93.17 ± 0.25	95.07 ± 0.27
Leaky ReLU(α = 0.01)	99.05 ± 0.14	93.12 ± 0.29	95.18 ± 0.23
PReLU	99.03±0.19	93.11±0.33	94.99±0.31
ReLU6	98.91±0.12	93.14±0.25	94.91±0.28
GELU	99.10±0.11	93.27±0.23	95.33±0.30
ELU	99.09 ± 0.15	93.08 ± 0.32	95.23 ± 0.30
Softplus	98.89 ± 0.12	92.89 ± 0.29	95.04 ± 0.34
Swish	99.14 ± 0.07	93.3 ± 0.19	95.31 ± 0.22
PAU	99.12±0.07	93.35±0.23	95.21 ±0.20
CP-1	99.10±0.15	93.20±0.31	95.14±0.30
CP-2	99.07±0.12	93.24±0.28	95.09±0.31
LAU	99.12±0.14	93.11±0.24	95.14±0.30
LEG	99.14±0.12	93.22±0.28	95.12±0.24
HP-1	**99.31**±0.08	**93.75**±0.17	**95.67**±0.18
HP-2	**99.27**±0.07	**93.74**±0.20	**95.61**±0.21

Table 7.3: Comparison between different baseline activations, HP-1, and HP-2 activations on MNIST, Fashion MNIST, and SVHN datasets on VGG-16 network. We report results for 10-fold mean accuracy (in %). mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.12±0.11	93.28±0.22	95.45±0.21
Leaky ReLU(α = 0.01)	99.11±0.14	93.24±0.25	95.41±0.27
PReLU	99.17±0.17	93.30±0.31	95.51±0.35
ReLU6	99.06±0.10	93.21±0.24	95.49±0.21
GELU	99.21±0.12	93.39±0.27	95.62±0.29
ELU	99.19±0.16	93.34±0.32	95.49±0.29
Softplus	99.04±0.14	93.12±0.31	95.22±0.37
Swish	99.25±0.13	93.36±0.22	95.67±0.21
PAU	99.22±0.11	93.39±0.21	95.52±0.19
CP-1	99.23±0.12	93.31±0.27	95.59±0.22
CP-2	99.20±0.11	93.41±0.24	95.49±0.23
LAU	99.22±0.13	93.57±0.27	95.42±0.27
LEG	99.28±0.12	93.34±0.28	95.45±0.24
HP-1	**99.57±0.09**	**93.88±0.21**	**95.94±0.18**
HP-2	**99.44±0.10**	**93.79±0.21**	**95.83±0.20**

Table 7.4: Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in AlexNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	98.92±0.12	91.01±0.24	93.12±0.19
Leaky ReLU(α = 0.01)	98.87±0.14	91.11±0.20	93.19±0.22
PReLU	99.01±0.19	91.15±0.25	93.27±0.20
ReLU6	99.00±0.09	91.12±0.20	93.24±0.19
GELU	99.12±0.15	91.27±0.21	93.42±0.20
ELU	99.01±0.12	91.22±0.24	93.35±0.22
Softplus	98.82±0.17	91.01±0.28	93.17±0.36
Swish	99.12±0.10	91.31±0.25	93.52±0.22
PAU	99.11±0.12	91.39±0.25	93.46±0.21
CP-1	99.15±0.14	91.36±0.28	93.41±0.30
CP-2	99.10±0.09	91.22±0.24	93.37±0.26
LAU	99.17±0.15	91.36±0.22	93.48±0.21
LEG	99.14±0.14	91.47±0.27	93.51±0.25
HP-1	**99.42±0.08**	**91.79±0.20**	**93.77±0.18**
HP-2	**99.36±0.10**	**91.72±0.24**	**93.82±0.21**

Table 7.5: Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in LeNet. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

Activation Function	MNIST	Fashion MNIST	SVHN
ReLU	99.14±0.09	92.87±0.21	95.17±0.24
Leaky ReLU(α = 0.01)	99.22±0.11	92.91±0.20	95.22±0.27
PReLU	99.17±0.15	92.81±0.29	95.21±0.31
ReLU6	99.11±0.10	92.85±0.22	95.12±0.23
GELU	99.19±0.11	92.99±0.25	95.25±0.26
ELU	99.15±0.11	92.97±0.31	95.19±0.27
Softplus	99.01 ±0.15	92.78±0.28	95.01±0.35
Swish	99.23±0.09	92.99±0.25	95.27±0.22
PAU	99.24±0.11	93.05±0.21	95.29±0.24
CP-1	99.22±0.14	93.02±0.22	95.25±0.27
CP-2	99.27±0.14	93.01±0.25	95.23±0.28
LAU	99.25±0.12	93.15±0.24	95.30±0.29
LEG	99.20±0.15	93.09±0.29	95.21±0.31
HP-1	**99.47±0.10**	**93.39±0.24**	**95.43±0.22**
HP-2	**99.40±0.09**	**93.31±0.21**	**95.42±0.25**

Table 7.6: Comparison between different baseline activations and safe OPAU activations on MNIST, Fashion MNIST, and SVHN datasets in Custom network. 10-fold mean accuracy (in %) have been reported. mean±std is reported in the table.

CIFAR

The CIFAR (Krizhevsky (2009)) database consists of total 60k images 32×32 RGB images, which is divided into 50k training and 10k test images. CIFAR dataset is divided into two database- CIFAR10 and CIFAR100. CIFAR10 database contains total 10 classes with 6000 images per class, while CIFAR100 database contains total 100 classes with 600 images per class. Experimental results for top-1 accuracy for the mean of 10 runs is reported on Table 7.7 and Table 7.8 on CIFAR10 dataset CIFAR100 datasets, respectively. Results are reported in both the database on VGG-16 (with batch-normalization) (Simonyan and Zisserman (2015)), PreActResNet-34 (PA-ResNet-34) (He *et al.* (2016)), Densenet-121 (DN-121) (Huang *et al.* (2016*a*)), MobileNet V2 (MN V2) (Sandler *et al.* (2019)), Shufflenet V2 (SF V2) (Ma *et al.* (2018)), EfficientNet B0 (EN-B0) (Tan and Le (2020)), and LeNet (Lecun *et al.* (1998)) networks. More experimental results on AlexNet (AN) (Krizhevsky *et al.* (2012)), Resnet-50 (He *et al.* (2015*a*)), Deep Layer Aggregation (DLA) (Yu *et al.* (2019)), Googlenet (Szegedy *et al.* (2014*a*)), Inception Network (Szegedy *et al.* (2015*b*)), WideresNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), Resnext (Xie *et al.* (2017)) are reported in Table 7.9 and 7.10. It evident from Table 7.7, 7.8, 7.9, and 7.10 that in most cases HP-1, and HP-2 constantly outperforms ReLU and Swish. Also, notice that there is an improvement in Top-1 accuracy from 1% to 6% when compared with ReLU activation in the above mentioned networks. We have considered batch size of 128, Adam optimizer Kingma and Ba (2015) with 0.001 learning rate and trained the networks up-to 100 epochs. Data augmentation is used for both the datasets. Analysing these learning curves, it is clear that after training few epochs HP-1, and HP-2 have faster convergence speed, smooth & stable learning, higher accuracy and lower loss when compared to ReLU. Experiments with AlexNet (with batchNorm) (Krizhevsky *et al.* (2012)), ResNet-50 (He *et al.* (2015*a*)), GoogleNet (GN) (Szegedy *et al.* (2014*a*)), Inception V3 (IN v3) (Szegedy *et al.* (2015*b*)), WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), ResNext-50 (RNxt-50) (Xie *et al.* (2017)), and Deep Layer Aggregation (DLA) Yu *et al.* (2019) is reported on Table 7.9, and 7.10 on CIFAR10 and CIFAR100 datasets respectively. Accuracy and loss curve on CIFAR10 dataset on LeNet Lecun *et al.* (1998) is given in figure 7.3 and 7.4. Training and validation curves for ReLU, Leaky ReLU, ELU, Softplus, Swish, PAU, CP-1, CP-2, LAU, LEG, HP-1, and HP-2 activations are given in Figures 7.5 and 7.6 on CIFAR100 dataset on

MobileNet V2 (Sandler *et al.* (2019)) network.

Activation Function	VGG-16	MN V2	PA-ResNet-34	SF V2	LeNet	DN-121	EN-B0
ReLU	89.78 ±0.18	89.71 ±0.30	90.18 ±0.21	88.47 ±.34	67.35 ±0.47	91.89 ±0.21	85.52 ±0.34
Leaky ReLU (α = 0.01)	89.89 ±0.21	89.92 ±0.31	90.32 ±0.24	88.59 ±0.31	66.99 ±0.54	92.15 ±0.23	85.42 ±0.37
PReLU	89.72 ± 0.32	89.79 ±0.39	90.32 ±0.35	88.61 ±0.41	67.18 ±0.29	91.81 ±0.31	85.41 ±0.41
ReLU6	89.76 ±0.25	89.67 ±0.31	90.15 ±0.19	88.57 ±0.34	67.31 ±0.32	91.99 ±0.24	85.27 ±0.32
GELU	89.99 ±0.26	90.12 ±0.31	90.42 ±0.24	88.41 ±0.32	67.59 ±0.24	92.07 ±0.29	86.12 ±0.31
ELU	89.97 ±0.27	89.62 ±0.34	90.20 ±0.31	88.69 ±0.39	67.02 ±0.57	92.07 ±0.31	85.65 ±0.41
Softplus	89.62 ±0.24	89.52 ±0.35	90.01 ±0.33	88.52 ±0.42	66.71 ±0.55	91.71 ±0.34	85.07 ±0.39
Swish	90.47 ±0.21	90.07 ±0.31	90.98 ±0.24	89.22 ±0.34	67.95 ±0.46	91.87 ±0.21	86.27 ±0.31
PAU	90.21 ±0.22	90.27 ±0.32	90.49 ±0.27	89.37 ±0.35	68.42 ±0.48	92.14 ±0.23	86.35 ±0.34
CP-1	90.27 ±0.27	90.35 ±0.29	91.28 ±0.29	89.67 ±0.39	68.30 ±0.28	91.36 ±0.24	86.66 ±0.37
CP-2	90.42 ±0.28	90.29 ±0.24	91.38 ±0.31	89.55 ±0.32	68.07 ±0.34	91.51 ±0.29	86.48 ±0.39
LAU	90.34 ±0.31	90.22 ±0.29	91.21 ±0.26	89.51 ±0.38	68.31 ±0.25	91.81 ±0.20	86.41 ±0.32
LEG	90.29 ±0.24	90.35 ±0.31	91.02 ±0.28	89.48 ±.37	68.40 ±0.29	91.89 ±0.25	86.23 ±0.29
HP-1	**91.24 ± 0.20**	**90.92 ±0.28**	**92.20 ±0.20**	**90.12 ±0.33**	**69.59 ±0.41**	**92.97 ±0.20**	**87.67 ±0.24**
HP-2	**91.12 ±0.20**	**90.67 ±0.32**	**91.96 ±0.22**	**89.92 ±0.36**	**69.41 ±0.44**	**92.77 ±0.21**	**87.55 ±0.28**

Table 7.7: Comparison between different baseline activations, HP-1, and HP-2 activations on CIFAR10 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table.

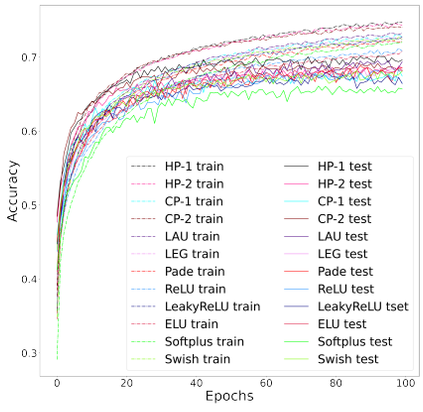

Figure 7.3: Top-1 Train and Test accuracy (higher is better) on CIFAR10 dataset with LeNet model for different activations

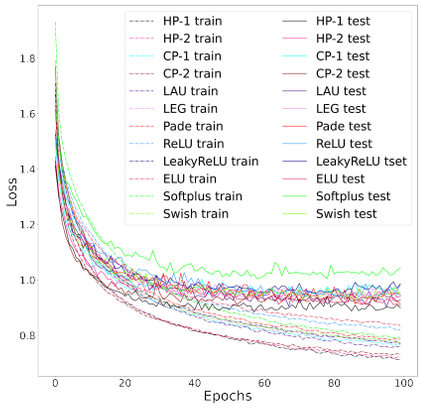

Figure 7.4: Top-1 Train and Test loss (lower is better) on CIFAR10 dataset with LeNet model for for different activations

Activation Function	AN	ResNet-50	GN	IN V3	WRN 28-10	RNxt-50	DLA
ReLU	89.12 ±0.29	90.65 ±0.19	91.01 ±0.21	91.67 ±0.17	91.81 ±0.23	91.42 ±0.19	89.59 ±0.27
Leaky ReLU (α = 0.01)	89.27 ±0.25	90.79 ±0.24	91.09 ±0.25	91.58 ±0.17	91.69 ±0.22	91.52 ±0.26	89.51 ±0.32
PReLU	89.34 ±0.29	90.69 ±0.27	91.17 ±0.23	91.77 ±0.24	91.77 ±0.27	91.57 ±0.35	89.61 ±0.29
ReLU6	89.22 ±0.31	90.75 ±0.20	91.01 ±0.25	91.59 ±0.19	91.63 ±0.25	91.35 ±0.19	89.52 ±0.26
GELU	89.65 ±0.31	90.82 ±0.29	90.91 ±0.24	92.01 ±0.24	92.07 ±0.18	91.77 ±0.19	90.19 ±0.25
ELU	89.49 ±0.27	90.41 ±0.27	91.21 ±0.29	91.97 ±0.26	91.89 ±0.25	91.47 ±0.19	89.88 ±0.32
Softplus	89.37 ±0.34	90.35 ±0.32	90.81 ±0.26	91.54 ±0.24	91.52 ±0.29	91.35 ±0.29	89.51 ±0.35
Swish	89.89 ±0.24	**90.99 ±0.18**	91.32 ±0.25	92.13 ±0.23	92.27 ±0.24	91.65 ±0.20	90.12 ±0.30
PAU	89.65 ±0.23	90.46 ±0.21	91.49 ±0.25	92.31 ±0.19	92.54 ±0.29	91.91 ±0.25	90.27 ±0.24
CP-1	89.55 ±0.29	90.50 ±0.23	91.45 ±0.29	92.34 ±0.24	92.42 ±0.34	91.81 ±0.23	90.37 ±0.36
CP-2	89.69 ±0.31	90.47 ±0.23	91.42 ±0.29	92.27 ±0.23	92.49 ±0.25	91.77 ±0.23	90.29 ±0.32
LAU	89.59 ±0.30	90.56 ±0.23	91.31 ±0.25	92.41 ±0.21	92.39 ±0.29	91.85 ±0.25	90.20 ±0.32
LEG	89.75 ±0.31	90.51 ±0.26	91.52 ±0.26	92. ±0.29	92.61 ±0.29	91.69 ±0.23	90.24 ±0.30
HP-1	**90.25 ±0.21**	90.59 ±0.21	**92.41 ±0.20**	**93.61 ±0.14**	**93.42 ±0.20**	**93.23 ±0.15**	**91.24 ±0.23**
HP-2	**90.40 ±0.23**	90.56 ±0.19	**92.25 ±0.24**	**93.50 ±0.16**	**93.29 ±0.20**	**93.01 ±0.14**	**91.10 ±0.32**

Table 7.8: Comparison between different baseline activations and safe OPAU activations on CIFAR10 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table.

Activation Function	AN	ResNet-50	GN	IN V3	WRN 28-10	RNxt-50	DLA
ReLU	59.49 ±0.40	64.52 ±0.41	70.18 ±0.32	69.34 ±0.39	69.79 ±0.31	69.11 ±0.43	63.15 ±0.39
Leaky ReLU (α = 0.01)	59.49 ±0.49	64.72 ±0.42	70.12 ±0.34	69.51 ±0.35	69.61 ±0.38	69.19 ±0.40	63.01 ±0.37
PReLU	59.65 ±0.45	64.65 ±0.43	70.29 ±0.38	69.37 ±0.40	69.82 ±0.40	69.29 ±0.42	63.25 ±0.42
ReLU6	59.35 ±0.39	64.57 ±0.42	70.01 ±0.35	69.42 ±0.38	69.69 ±0.35	69.34 ±0.45	63.22 ±0.42
GELU	59.99 ±0.48	65.29 ±0.40	70.68 ±0.35	70.29 ±0.39	69.84 ±.39	69.59 ±0.40	63.35 ±0.47
ELU	59.41 ±0.40	64.41 ±0.47	70.51 ±0.42	69.21 ±0.45	69.65 ±0.36	69.11 ±0.47	63.39 ±0.46
Softplus	59.19 ±0.45	64.59 ±0.45	70.00 ±0.39	69.24 ±0.51	69.52 ±0.39	68.92 ±0.44	63.01 ±0.48
Swish	60.35 ±0.38	**65.85 ±0.41**	70.85 ±0.40	70.21 ±0.35	70.19 ±0.39	70.12 ±0.35	63.52 ±0.39
PAU	60.65 ±0.42	65.65 ±0.41	71.12 ±0.36	70.39 ±0.38	70.52 ±0.42	70.01 ±0.45	63.82 ±0.40
CP-1	60.40 ±0.48	65.21 ±0.40	71.22 ±0.39	70.31 ±0.37	70.29 ±0.45	70.19 ±00.49	63.41 ±0.49
CP-2	60.54 ±0.45	65.32 ±0.45	71.25 ±0.39	70.11 ±0.42	70.37 ±0.39	70.31 ±0.44	63.56 ±0.51
LAU	60.32 ±0.40	65.39 ±0.45	71.11 ±0.40	70.51 ±0.39	70.21 ±0.45	70.09 ±0.44	63.25 ±0.45
LEG	60.39 ±0.45	65.19 ±0.45	71.39 ±0.41	70.44 ±036	70.51 ±0.44	69.99 ±0.49	63.37 ±0.46
HP-1	**61.20 ±0.38**	65.45 ±0.42	**72.09 ±0.30**	**71.67 ±0.35**	**71.62 ±0.31**	**71.27 ±0.40**	**64.22 ±0.31**
HP-2	**61.01 ±0.35**	65.39 ±0.32	**72.36 ±0.28**	**71.35 ±0.31**	**71.47 ±0.25**	**70.92 ±0.35**	**64.51 ±0.38**

Table 7.9: Comparison between different baseline activations and safe OPAU activations on CIFAR100 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table.

Tiny Imagenet

The ImageNet Large Scale Visual Recognition Challenge(ILSVRC) is considered to be one of the most popular benchmarks for image classification problems. A similar type of image classification database like ILSVRC is Tiny Imagenet, which is a smaller dataset with fewer image classes. The images in this database are of size 64×64 with total 100,000 training images, 10,000 validation images, and 10,000 test images. The database has 200 image classes with 500 training images, 50 validation images, and 50 test images in each class. A mean of 5 different runs for Top-1 accuracy is reported in table 7.11 on WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)) network. We have used a batch size of 32, He Normal initializer (He *et al.* (2015*b*)), 0.2 dropout rate (Srivastava *et al.* (2014)), adam optimizer (Kingma and Ba (2015)), initial

Activation Function	VGG-16	MN V2	PA-ResNet-34	SF V2	LeNet	DN-121	EN-B0
ReLU	57.03 ±0.62	63.20 ±0.60	60.39 ±0.51	61.30 ±0.47	32.62 ±0.35	67.50 ±0.38	53.02 ±0.49
Leaky ReLU (α = 0.01)	57.17 ±0.59	63.54 ±0.62	60.51 ±0.50	61.55 ±0.39	32.94 ±0.35	67.61 ±0.42	53.25 ±0.44
PReLU	57.15 ±0.67	63.35 ±0.51	60.15 ±0.59	61.12 ±0.54	33.12 ±0.48	67.62 ±0.45	53.15 ±0.54
ReLU6	57.19 ±0.64	63.22 ±0.48	60.01 ±0.54	61.35 ±0.48	33.29 ±0.45	67.50 ±0.39	53.17 ±0.55
GELU	57.59 ±0.57	63.82 ±0.57	62.20 ±0.58	62.01 ±0.54	33.12 ±0.38	67.81 ±0.57	53.59 ±0.54
ELU	57.59 ±0.71	63.47 ±0.64	60.89 ±0.47	61.85 ±0.52	33.89 ±0.31	67.32 ±0.49	53.34 ±0.45
Softplus	56.89 ±0.70	63.28 ±0.70	60.32 ±0.61	61.22 ±0.59	32.84 ±0.45	67.42 ±0.38	53.17 ±0.41
Swish	58.22 ±0.55	63.91 ±0.56	62.02 ±0.48	62.26 ±.47	34.12 ±0.39	68.07 ±0.32	54.54 ±0.34
PAU	58.54 ±0.58	64.97 ±0.51	62.18 ±0.53	62.14 ±0.51	33.94 ±0.35	67.96 ±0.41	53.81 ±0.40
CP-1	58.31 ±0.70	65.45 ±0.57	64.34 ±0.54	62.34 ±0.49	34.57 ±0.36	68.32 ±0.45	54.35 ±0.51
CP-2	58.22 ±0.64	65.32 ±0.45	64.49 ±0.55	62.17 ±0.55	34.18 ±0.39	68.17 ±0.38	54.28 ±0.39
LAU	58.54 ±0.61	65.64 ±0.45	64.01 ±0.49	62.12 ±0.55	34.01 ±0.48	68.09 ±0.45	54.21 ±0.48
LEG	58.19 ±0.64	65.51 ±0.45	64.65 ±0.51	62.29 ±0.50	34.21 ±0.36	67.89 ±0.39	54.54 ±0.55
HP-1	**60.77 ±0.52**	**66.22 ±0.55**	**65.45 ±0.49**	**63.04 ±0.42**	**35.36 ±0.21**	**68.72 ±0.30**	**54.99 ±0.37**
HP-2	**60.64 ±0.58**	**65.95 ±0.51**	**65.02 ±0.54**	**63.02 ±0.46**	**34.85 ±0.27**	**68.66 ±0.27**	**54.74 ±0.42**

Table 7.10: Comparison between different baseline activations, HP-1, and HP-2 activations on CIFAR100 dataset. We report results for Top-1 accuracy(in %) for mean of 10 different runs. mean±std is reported in the table.

learning rate(lr rate) 0.01, and lr rate is reduced by a factor of 10 after every 50 epochs up-to 250 epochs. The Data augmentation method is used in this database.

7.7.2 Object Detection

Object Detection is considered one of the most important problems in computer vision. The Pascal VOC dataset (Everingham *et al.* (2010)) is used for our object detection experiments. We report results on Single Shot MultiBox Detector(SSD) 300 (Liu *et al.* (2016)) with VGG-16(with batch-normalization) as the backbone network. We do not use any pre-trained weight in the network. The network is trained on Pascal VOC 07+12 training data and tested network performance on Pascal VOC 2007 test data. We consider a batch size of 8, 0.001 learning rate, SGD optimizer (Robbins and Monro (1951);

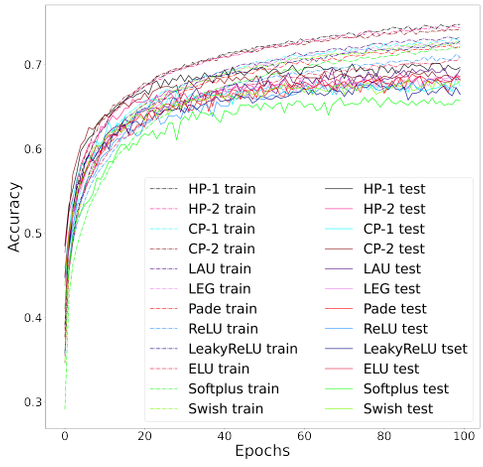

Figure 7.5: Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with MobileNet V2 network for different activations

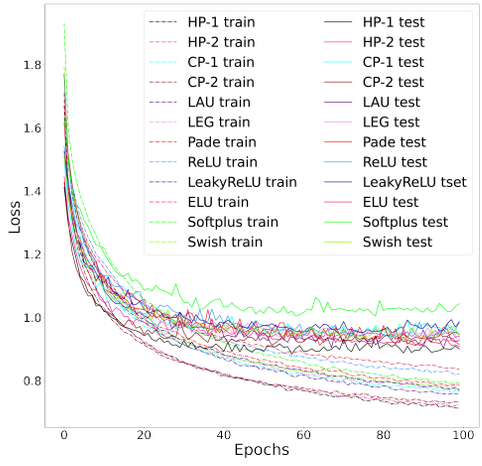

Figure 7.6: Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with MobileNet V2 network for different activations

Kiefer and Wolfowitz (1952)) with 0.9 momentum, $5e^{-4}$ weight decay for 120000 iterations. We report results for the mean average precision(mAP) in Table 7.12 for a mean of 5 different runs.

7.7.3 Semantic Segmentation

Semantic segmentation another important problem in computer vision. We report experimental results on the Cityscapes dataset (Cordts *et al.* (2016)). We use U-net network (Ronneberger *et al.* (2015)) as the segmentation framework and is trained for 250 epochs, with adam optimizer (Kingma and Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)). A mean of 5 different runs for Pixel Accuracy and mean Intersection-Over-Union (mIOU) on test data is reported on table 7.13.

7.7.4 Machine Translation

Machine Translation is a deep learning technique in which one language is translated to another language. For our experiments, WMT 2014 English→German dataset is used. The database contains 4.5 million training sentences. We evaluate network performance on the newstest2014 dataset using the BLEU score metric. An Attention-based 8-head

Activation Function	Top-1 accuracy(in %)
ReLU	60.35±0.53
Leaky ReLU(α = 0.01)	60.62±0.42
PReLU	60.21±0.67
ReLU6	60.47±0.54
GELU	60.91±0.59
ELU	60.02±0.67
Softplus	59.81±0.61
Swish	60.69±0.42
PAU	60.52±0.47
CP-1	60.54±0.61
CP-2	60.61±0.49
LAU	60.32±0.57
LEG	61.31±0.64
HP-1	**62.52±0.39**
HP-2	**62.21±0.46**

Table 7.11: Comparison between different baseline activations, HP-1, and HP-2 activations on the Image classification Problem. We report results for mean of 5 different runs on WRN 28-10 network on Tiny Imagenet Dataset. mean±std is reported in the table.

Activation Function	mAP
ReLU	77.2±0.14
Leaky ReLU(α = 0.01)	77.2±0.19
PReLU	77.2±0.21
ReLU6	77.1±0.14
GELU	77.3±0.19
ELU	75.1±0.22
Softplus	74.2±0.25
Swish	77.3±0.11
PAU	77.3±0.15
CP-1	77.3±0.17
CP-2	77.3±0.15
LAU	77.2±0.19
LEG	77.4±0.12
HP-1	**78.0±0.14**
HP-2	**77.9±0.10**

Table 7.12: Comparison between different baseline activations, HP-1, and HP-2 activations on the Object Detection Problem. We report results on SSD 300 with VGG-16 backbones on Pascal-VOC dataset. mean±std is reported in the table.

transformer network (Vaswani *et al.* (2017)) is used with Adam optimizer (Kingma and Ba (2015)), 0.1 dropout rate (Srivastava *et al.* (2014)), and trained up to 100000 steps. Other hyper-parameters are tried to keep similar as mentioned in the original paper (Vaswani *et al.* (2017)). Mean of 5 runs is been reported on Table 7.14 on the test dataset(newstest2014).

7.8 Comparison With the baseline activation functions

We observe that HP-1 and HP-2, in most cases, beat or performs equally well with baseline activation functions and under-performs marginally on rare occasions, and a detailed comparison of these activation functions on the basis of all the experiments provided in earlier sections is given in Table 7.15.

Activation Function	Pixel Accuracy	mIOU
ReLU	79.49±0.41	69.10±0.32
Leaky ReLU (α = 0.01)	79.51±0.5	69.22±0.39
PReLU	79.66±0.59	69.08±0.41
ReLU6	79.41±0.48	69.01±0.35
GELU	79.95±0.39	69.89±0.30
ELU	79.45±0.59	69.18±0.47
Softplus	78.78±0.65	68.01±0.52
Swish	79.97±0.41	69.45±0.27
PAU	79.92±0.49	69.62±0.34
CP-1	80.01±0.49	69.85±0.39
CP-2	80.17±0.52	70.19±0.40
LAU	70.91±0.41	69.99±0.32
LEG	80.01±0.48	70.12±0.37
HP-1	**81.22±0.37**	**70.72±0.29**
HP-2	**81.09±0.45**	**70.59±0.32**

Table 7.13: Comparison between different baseline activations, HP-1, and HP-2 activations on semantic segmentation Problem. We report results on U-NET network on the Cityscapes dataset. mean±std is reported in the table.

Activation Function	BLEU Score
ReLU	26.2±0.15
Leaky ReLU (α = 0.01)	26.3±0.17
PReLU	26.2±0.21
ReLU6	26.1±0.14
GELU	26.4±0.19
ELU	25.1±0.15
Softplus	23.6±0.16
Swish	26.4±0.10
PAU	26.2±0.12
CP-1	26.4±0.18
CP-2	26.3±0.14
LAU	26.3±0.20
LEG	26.3±0.20
HP-1	**26.8±0.10**
HP-2	**26.7±0.12**

Table 7.14: Comparison between different baseline activations, HP-1, and HP-2 activations on Machine translation Problem. We report results on Multi-head transformer network on the WMT-2014 dataset. mean±std is reported in the table.

Baselines	ReLU	Leaky ReLU	ELU	Softplus	PReLU	ReLU6	GELU	Swish	PAU
HP-1 $>$ Baseline	43	43	44	44	43	43	43	42	43
HP-1 $=$ Baseline	0	0	0	0	0	0	0	0	0
HP-1 $<$ Baseline	1	1	0	0	1	1	1	2	1
HP-2 $>$ Baseline	43	43	44	44	43	43	43	42	43
HP-2 $=$ Baseline	0	0	0	0	0	0	0	0	0
HP-2 $<$ Baseline	1	1	0	0	1	1	1	2	1

Table 7.15: Baseline table for HP-1 and HP-2. These numbers represent the total number of networks in which HP-1 and HP-2 outperforms, equal or underperforms when we compare with the baseline activation functions

7.9 Computational Time Comparison

HP-1, HP-2 contains trainable parameters, which increases the complexity of the network (like PAU), and due to this, there is a trade-off between network performance and computational cost. We have reported the Computational time comparison for HP-1,

HP-2, and the baseline activation functions for both forward and backward pass on a 32×32 RGB image on VGG-16 model in Table 7.16 for the mean of 100 runs. We have used an NVIDIA Tesla V100 GPU with 32GB ram.

Activation Function	Forward Pass(STD)	Backward Pass(STD)
ReLU	5.18($\pm$1.10) μs	5.55($\pm$1.24) μs
Swish	6.26($\pm$0.97) μs	8.21($\pm$1.57) μs
Leaky ReLU (α = 0.01)	5.26($\pm$1.05) μs	5.67($\pm$1.48) μs
ELU	5.56($\pm$1.12) μs	5.77($\pm$1.28) μs
Softplus	5.50($\pm$1.25) μs	5.41($\pm$1.37) μs
GELU	8.01($\pm$1.90) μs	8.23($\pm$2.54) μs
ReLU6	5.30($\pm$1.20) μs	5.81($\pm$1.21) μs
PReLU	5.62($\pm$1.22) μs	5.92($\pm$1.35) μs
PAU	13.42($\pm$2.12)μs	23.22($\pm$3.01)μs
HP-1	14.50($\pm$2.89) μs	20.55($\pm$2.13) μs
HP-2	14.65($\pm$2.67) μs	20.86($\pm$2.20) μs

Table 7.16: Runtime comparison for the forward and backward passes for HP-1, HP-2, and baseline activation functions for a 32$\times$ 32 RGB image in VGG-16 model.

7.10 Conclusion

In this chapter, two trainable and effective activation functions have been proposed, which are called HP-1 and HP-2, based on rational function approximation. From the experimental evaluation, it is clear that replacing known activation functions like PAU, ReLU, Swish etc., with HP-1 and HP-2 provides significant improvement in results on important deep learning problems like image classification, object detection, semantic segmentation, and machine translation.

A significant drawback of the proposed functions is that they have higher running times compared to ReLU, Swish, and GELU and have almost similar running times to PAU. The proposed activation functions contain ten trainable parameters, which increases the computational cost. A future direction can be to decrease the trainable parameters in the proposed functions to reduce the computational cost.

CHAPTER 8

TanhSoft [1]

8.1 Introduction

Artificial neural networks (ANNs) have occupied the center stage in deep learning in the recent past. ANNs are made up of several hidden layers, and each hidden layer consists of several neurons. At each neuron, an affine linear map is composed of a nonlinear function known as *activation function*. During the training of an ANN, the linear map is optimized; however, an activation function is usually fixed in the beginning, along with the architecture of the ANN. There has been an increasing interest in developing a methodical understanding of activation functions, particularly with regard to the construction of novel activation functions and identifying mathematical properties leading to better learning (Nwankpa *et al.* (2018)). An activation function is considered good if it can generalise better on a variety of datasets, ensure faster convergence and improve neural network performance, which leads to more accurate results. At the early stage of deep learning research, researchers used shallow networks (fewer hidden layers), along with tanh or sigmoid were used as activation functions. As the research progressed and deeper networks (multiple hidden layers) came into fashion to achieve challenging tasks, the Rectified Linear Unit (ReLU) (Nair and Hinton (2010); Hahnloser *et al.* (2000)) emerged as the most popular activation function. Despite its simplicity, deep neural networks with ReLU have learned many complex and highly nonlinear functions with high accuracy. To overcome the shortcomings of ReLU (non-zero mean, negative missing, unbounded output, to name a few (Zhou *et al.* (2020)), and to increase the accuracy considerably in a variety of tasks in comparison to networks with ReLU, many new activation functions have been proposed over the years. Many of these new activation functions are variants of ReLU, for example, Leaky ReLU (Maas *et al.* (2013*a*)), Exponential Linear Unit (ELU)) (Clevert *et al.* (2016)), Parametric Rectified Linear Unit (PReLU) (He *et al.* (2015*b*)), Randomized Leaky Rectified

[1] This chapter is a slightly modified version of the paper accepted at IEEE access Biswas *et al.* (2021*e*).

Linear Units (RReLU) (Xu *et al.* (2015*a*)), and Inverse Square Root Linear Units (ISRLU) (Carlile *et al.* (2017)), Flexible ReLU (FReLU) (Qiu *et al.* (2017)). In the recent past, some activation functions constructed from tanh or sigmoid have achieved state-of-the-art results on a variety of challenging datasets. Most notably, among such activation functions, Swish Ramachandran *et al.* (2017) has emerged as a close favourite to ReLU. Some of these novel activation functions have shown that introducing hyper-parameters in the argument of the functions may provide activation functions for special values of these hyper-parameters that can outperform activation functions for other values of hyper-parameters, for example, see (Ramachandran *et al.* (2017); Zhou *et al.* (2020)).

8.2 Related works and Motivation

An activation function that can improve neural network model performance is an active field of research. It is always hard to find the best activation function. In earlier days, Tanh and Sigmoid were mostly used as activation functions in networks. ReLU (Nair and Hinton (2010)) was first proposed by Nair and Hinton in 2010, and since then, ReLU has been the widely used activation function in neural network models due to its simplicity. ReLU produces a positive outcome for positive inputs. In contrast, zero for negative inputs, and due to this, ReLU undergoes from vanishing gradient problem, which is known as dying ReLU (Maas *et al.* (2013*a*)). Several activation functions have been suggested by researchers to overcome this problem. Leaky ReLU (Maas *et al.* (2013*a*)) has been proposed with a small negative linear function for negative input, and it shows promising results compared to ReLU. PReLU (He *et al.* (2015*b*)) has been introduced with a modification of Leaky ReLU and added a trainable linear part for negative inputs. Later, RReLU (Xu *et al.* (2015*b*)), ISReLU (Carlile *et al.* (2017)), FReLU (Qiu *et al.* (2017)), PELU (Trottier *et al.* (2017)), SiLU (Elfwing *et al.* (2017)), ELU (Clevert *et al.* (2016)), and GELU (Hendrycks and Gimpel (2020)) have been proposed and they have improved model performances. Mish (Misra (2020)), which has been introduced recently, has shown some improvement over ReLU and Swish (Ramachandran *et al.* (2017)). Most of the functions mentioned above, except PReLU, PELU, and FReLU are non-trainable. A trainable activation function contains trainable parameter(s), which are tuned via backpropagation. In the early 1990s and 2000s, during the pre-ReLU era, there were a few trainable activations proposed like Adjustable Gener-

alized Hyperbolic Tangent (Chen and Chang (1996)), Sigmoidal selector (Singh and Chandra (2003)) etc. Later, Leaky ReLU, ELU, and ReLU were modified by PReLU, PELU, and FReLU, respectively, by introducing trainable parameter(s). Recently, in 2017, Swish (Ramachandran *et al.* (2017)), a trainable activation, was found using exhaustive search (Negrinho and Gordon (2017)), and reinforcement learning techniques (Baker *et al.* (2016)), which found lots of attention from the deep learning community due to its simplicity and efficiency.

8.3 Research contribution

The standard ANN training process involves tuning the weights in the linear part of the network; however, there is merit in the ability to custom design activation functions, to better fit the problem at hand. Real-world datasets are noisy or challenging, and it is always difficult to construct the best activation function to generalize on random datasets. It is hard to say whether an activation function will generalize successfully and replace ReLU on challenging or noisy datasets. Though there may be merit in having a custom activation function corresponding to the problem at hand, but yet it is beneficial to identify activation functions that generalize to several real-world data sets, making it easier to implement. Hence we proposed three activation functions, namely, TanhSoft-1, TanhSoft-2, and TanhSoft-3, and established their generalizability and usefulness over other conventional activation functions. In what follows, we discuss the properties of these activations, experiments with complex models, and a comparison with a few other widely used activation functions.

8.4 TanhSoft-1, TanhSoft-2, and TanhSoft-3 & their properties

In this chapter, we propose three activation function with hyper-parameters which we call TanhSoft-1, TanhSoft-2, and TanhSoft-3. The hyper-parameters can be used as

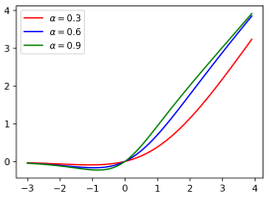

Figure 8.1: Plots of $\mathcal{F}_1(x;\alpha)$ for different values of α.

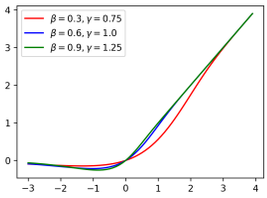

Figure 8.2: Plots of $\mathcal{F}_2(x;\beta,\gamma)$ for different values of β,γ.

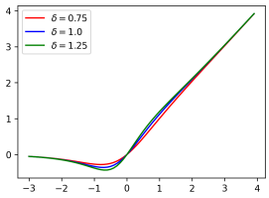

Figure 8.3: Plots of $\mathcal{F}_3(x;\delta)$ for different values of δ.

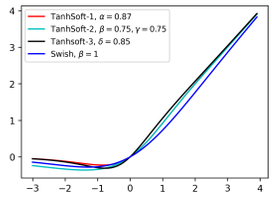

Figure 8.4: Plots of $\mathcal{F}_1(x;0.87)$, $\mathcal{F}_2(x;0.75,0.75)$, $\mathcal{F}_3(x;0.85)$ and Swish.

trainable or constant. TanhSoft-1, TanhSoft-2, and TanhSoft-3 are defined as

$$\begin{aligned}
\text{TanhSoft-1}: \quad \mathcal{F}_1(x;\alpha) &:= \tanh(\alpha x)\text{Softplus}(x) \\
&= \tanh(\alpha x)\ln(1+e^x), \qquad (8.1)\\
\text{TanhSoft-2}: \quad \mathcal{F}_2(x;\beta,\gamma) &:= x\tanh(\beta e^{\gamma x}), \qquad (8.2)\\
\text{TanhSoft-3}: \quad \mathcal{F}_3(x;\delta) &:= \ln(1+e^x\tanh(\delta x)). \qquad (8.3)
\end{aligned}$$

The corresponding derivatives are

$$\begin{aligned}
\frac{d}{dx}\mathcal{F}_1(x;\alpha) &= \tanh(\alpha x)\frac{e^x}{(1+e^x)} \\
&\quad +\alpha\,\text{sech}^2(\alpha x)\ln(1+e^x),
\end{aligned} \qquad (8.4)$$

$$\begin{aligned}
\frac{d}{dx}\mathcal{F}_2(x;\beta,\gamma) &= \tanh(\beta e^{\gamma x}) \\
&\quad +\beta\gamma x e^{\gamma x}\,\text{sech}^2(\beta e^{\gamma x}), \qquad (8.5)\\
\frac{d}{dx}\mathcal{F}_3(x;\delta) &= \frac{e^x\tanh(\delta x)+\delta e^x\,\text{sech}^2(\delta x)}{1+e^x\tanh(\delta x)}. \qquad (8.6)
\end{aligned}$$

Figures 8.1, 8.2 and 8.3 show the graph for $\mathcal{F}_1(x;\alpha)$, $\mathcal{F}_2(x;\beta,\gamma)$, and $\mathcal{F}_3(x;\delta)$ activation functions for different values of α and β,γ and δ respectively. Plots of the first derivative of $\mathcal{F}_1(x;\alpha)$, $\mathcal{F}_2(x;\beta,\gamma)$, and $\mathcal{F}_3(x;\delta)$ are given in Figures 8.5, 8.6, and 8.7 for different values of α and β,γ and δ respectively. A comparison between $\mathcal{F}_1(x;\alpha)$,

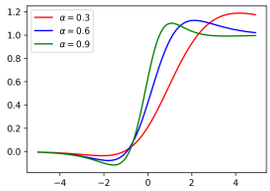

Figure 8.5: Plots of first derivative of $\mathcal{F}_1(x;\alpha)$ for different values of α.

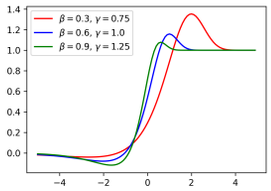

Figure 8.6: Plots of first derivative of $\mathcal{F}_2(x;\beta,\gamma)$ for different values of β,γ.

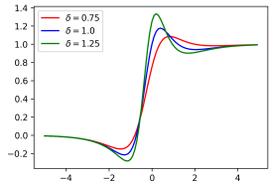

Figure 8.7: Plots of first derivative of $\mathcal{F}_3(x;\delta)$ for different values of δ.

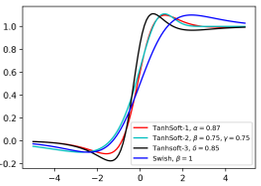

Figure 8.8: Plots of first order derivatives of $\mathcal{F}_1(x;0.87)$, $\mathcal{F}_2(x;0.75,0.75)$, $\mathcal{F}_3(x;0.85)$ and Swish.

$\mathcal{F}_2(x;\beta,\gamma)$, $\mathcal{F}_3(x;\delta)$ and Swish and their first derivatives are given in Figures 8.4 and 8.8. The author of (Misra (2020)) has reported unstable training behaviour for a specific function which can be obtained from TanhSoft-1, however, we tested and failed to find any such instability. Also, in (Liu and Di (2020)) the authors have mentioned a special case which can be obtained from TanhSoft-2.

The three functions have non-monotonic curvature in the negative axis. The hyper-parameters α for TanhSoft-1, β,γ for TanhSoft-2 and δ for TanhSoft-3 plays a major role and controls the slope of the curve in both positive and negative axes as evident from Figures 8.1, 8.2 and 8.3. Like Swish, $\mathcal{F}_1(x;\alpha)$, $\mathcal{F}_2(x;\beta,\gamma)$, and $\mathcal{F}_3(x;\delta)$ are both smooth, non-monotonic activation functions and bounded below.

$\mathcal{F}_1(x;\alpha)$, $\mathcal{F}_2(x;\beta,\gamma)$, and $\mathcal{F}_3(x;\delta)$ becomes the zero function for $\alpha=0,\beta=0$ and $\delta=0$ respectively. $\mathcal{F}_2(x;\beta,0)$ becomes the linear function family $\tanh(\beta)x$. For large values of some parameter while fixing the other parameters, the proposed functions converges to some known activation functions point-wise. For example,

$$\lim_{\gamma\to\infty}\mathcal{F}_2(x;\beta,\gamma) = \mathrm{ReLU}(x), \quad \forall x\in\mathbb{R} \text{ for any fixed } \beta>0. \tag{8.7}$$

Also, The class of neural networks with TanhSoft-1 or TanhSoft-2 or TanhSoft-3 activation function is dense in $C(K)$, where K is a compact subset of $\mathbb{R}^n$ and $C(K)$ is the space of all continuous functions over K (see (Molina *et al.* (2020))).

The proof follows from the following proposition as all three proposed activations are non-polynomial.

Proposition (Theorem 1.1 in Kidger and Lyons, 2019 Kidger and Lyons (2020)) :- Let $\rho : \mathbb{R} \rightarrow \mathbb{R}$ be any continuous function. Let N_n^ρ represent the class of neural networks with activation function ρ, with n neurons in the input layer, one neuron in the output layer, and one hidden layer with an arbitrary number of neurons. Let $K \subseteq \mathbb{R}^n$ be compact. Then N_n^ρ is dense in $C(K)$ if and only if ρ is non-polynomial.

8.5 Experiments with TanhSoft-1, TanhSoft-2, and TanhSoft-3

In this work we have initialized and updated hyper-parameter values of TanhSoft-1, TanhSoft-2, and TanhSoft-3 using the backpropagation (LeCun *et al.* (1989)) algorithm (see He *et al.* (2015*b*)). For a single layer, the gradient of a hyper-parameter θ is:

$$\frac{\partial E}{\partial \theta} = \sum_x \frac{\partial E}{\partial F(x)} \frac{\partial F(x)}{\partial \theta} \tag{8.8}$$

where E is the objective function, $\theta \in \{\alpha, \beta, \gamma, \delta\}$ and $F(x) \in \{\mathcal{F}_1(x; \alpha), \mathcal{F}_2(x; \beta, \gamma), \mathcal{F}_3(x; \delta)\}$.

We have considered several models and datasets to measure the performance of $\mathcal{F}_1(x; \alpha)$, $\mathcal{F}_2(x; \beta, \gamma)$, and $\mathcal{F}_3(x; \delta)$ and have compared with seven baseline widely used activation functions.

In the following subsections, we have provided experimental results of TanhSoft-1, TanhSoft-2, and TanhSoft-3 with baseline activation functions such as ReLU, Leaky ReLU, ELU, Softplus, Swish, GELU, and Mish for different deep learning problems like Image classification, Object detection, Semantic segmentation, and Machine translation. We have initialized $\alpha = 0.87$ for TanhSoft-1, $\beta = 0.75, \gamma = 0.75$ for TanhSoft-2, and $\delta = 0.85$ for TanhSoft-3 (see (He *et al.* (2015*b*))) and updated these hyper-parameter values via backpropagation during training as mentioned in equation (8). In the following subsections, we will provide details of our experimental setup, framework, and results. All the experiments were conducted on an NVIDIA tesla V-100 GPU with 16GB RAM.

8.5.1 Image Classification:

We have reported results for image classification problem in some widely used standard datasets like MNIST (LeCun *et al.* (2010)), Fashion MNIST (Xiao *et al.* (2017)), SVHN (Netzer *et al.* (2011)), CIFAR10 (Krizhevsky (2009)), CIFAR100 (Krizhevsky (2009)), and Tiny Imagenet (Le and Yang (2015)).

MNIST

The MNIST (LeCun *et al.* (2010)) database contains image data of handwritten digits from 0 to 9. The dataset contains 60k training and 10k testing 28×28 grey-scale images. A 8-layer customised homogeneous convolutional neural network (CNN) architecture with 3×3 kernels for CNN layers and 2×2 kernels for pooling layers are being used. We have used channel depths of size 128 (twice), 64 (thrice), 32 (twice), a dense layer of size 128, Max-pooling layer(thrice), batch-normalization (Ioffe and Szegedy (2015)), and dropout (Srivastava *et al.* (2014)) on the custom CNN architecture. Data augmentation method is not used. The results are reported in Table 8.1.

Activation Function	5-fold mean accuracy (%) on MNIST test data
TanhSoft-1	**99.40**
TanhSoft-2	**99.34**
TanhSoft-3	**99.37**
ReLU	99.14
Swish	99.18
Leaky ReLU($\alpha = 0.01$)	99.20
ELU	99.10
Softplus	99.05
Mish	99.28
GELU	99.21

Table 8.1: Experimental results on MNIST dataset.

Fashion MNIST

Fashion-MNIST (Xiao *et al.* (2017)) is a popular computer vision database consisting of 28×28 pixels grey-scale images, consists of ten fashion items in each class.

Activation Function	5-fold mean accuracy (%) on Fashion MNIST test data
TanhSoft-1	**93.52**
TanhSoft-2	**93.40**
TanhSoft-3	**93.37**
ReLU	92.90
Swish	92.92
Leaky ReLU(α = 0.01)	92.95
ELU	92.85
Softplus	92.40
Mish	93.17
GELU	93.12

Table 8.2: Experimental results on Fashion MNIST dataset.

It has 60k training images and 10k testing images. Fashion-MNIST provides a more challenging classification problem than MNIST. The data augmentation method is not used. We have considered the same CNN architecture used in the MNIST dataset for this database as well for training and testing purpose and, the results are reported in Table 8.2.

The Street View House Numbers (SVHN) Database

SVHN (Netzer *et al.* (2011)) is a popular image database consists of real-world house numbers of Google's street view images with 32×32 RGB images. The database has 73257 training images and 26032 testing images. The database has a total of 10 classes. We have considered the same CNN architecture used in the MNIST dataset for this database as well for training and testing purpose and, the results are reported in Table 8.3. We have used the data augmentation method in this database.

Activation Function	5-fold mean accuracy (%) on SVHN test data
TanhSoft-1	**95.36**
TanhSoft-2	**95.52**
TanhSoft-3	**95.43**
ReLU	95.14
Swish	95.23
Leaky ReLU(α = 0.01)	95.20
ELU	95.15
Softplus	95.08
Mish	95.33
GELU	95.20

Table 8.3: Experimental results on SVHN dataset.

CIFAR

The CIFAR (Krizhevsky (2009)) is a popular computer vision dataset consists of 32×32 colored images, with total 60k images and divided into 50k training and 10k test images. There are two type of CIFAR dataset - CIFAR10 and CIFAR100. CIFAR10 dataset has 10 classes with 6000 images per class while CIFAR100 has 100 classes with 600 images per class. Top-1 accuracy for mean of 10 runs have been reported on CIFAR10 dataset in Table 8.4 & 8.5 and on CIFAR100 dataset in Table 8.6 & 8.7 on ResNet-34 (He *et al.* (2015*a*)), PreActResNet-34 (PA-ResNet-34) (He *et al.* (2016)), VGG-16 (with Batch-normalization) (Simonyan and Zisserman (2015)), Densenet-121 (DN-121) (Huang *et al.* (2016*a*)), DenseNet-169 (DN-169) (Huang *et al.* (2016*a*)), InceptionNet V3 (IN-V3) (Szegedy *et al.* (2015*a*)), SimpleNet(SN) (Hasanpour *et al.* (2016)), MobileNet V2 (MN) (Sandler *et al.* (2019)), WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), GoogleNet (GN) (Szegedy *et al.* (2014*b*)), ResNeXt-50 (Xie *et al.* (2017)), StochasticDepth (Huang *et al.* (2016*b*)), Shufflenet V2 (Ma *et al.* (2018)), Deep Layer Aggregation (DLA) (Yu *et al.* (2019)), RegNet (Radosavovic *et al.* (2020)), NASNet (Zoph *et al.* (2018)), Resnet in Resnet (RIR) (Targ *et al.* (2016)), Xception Network (Chollet (2017)), EfficientNet B0 (EN-B0) (Tan

and Le (2020)), Le-Net (Lecun *et al.* (1998)) and SqueezeNet (SQ-Net) (Iandola *et al.* (2016)) models. It is clear from these tables that TanhSoft-1, TanhSoft-2, and TanhSoft-3 constantly outperforms ReLU and Swish in most cases and we have got around 1% to 6% improvement in Top-1 accuracy when compared to models with ReLU activation. We have trained the networks with batch size 128, Adam optimizer (Kingma and Ba (2015)) with 0.001 learning rate and up to 100 epochs for all the models mentioned above except SimpleNet and VGG-16, which is trained till 200 epochs. Data augmentation is used for both datasets. Learning curves of ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3 activations are given in Figures 8.9 & 8.10 on WideResNet 28-10 model in CIFAR100 dataset and Figures 8.11 & 8.12 on Le-net model in CIFAR10 dataset. From these figures it is evident that after training few epochs TanhSoft-1, TanhSoft-2, TanhSoft-3 have faster convergence capability, higher accuracy and lower loss when compared to ReLU.

Activation Function	VGG-16	WRN 28-10	ResNet-34	PA-ResNet-34	DN-121	DN-169	IN-V3	MN-V2	SN	SQ-Net
TanhSoft-1	**90.77**	**93.42**	**91.98**	**92.22**	**93.69**	**93.17**	**93.12**	**91.55**	92.12	**88.12**
TanhSoft-2	**90.70**	**93.67**	**91.77**	**92.53**	**93.97**	**93.32**	**93.07**	**91.87**	92.21	**88.01**
TanhSoft-3	**90.97**	**93.18**	**91.89**	**92.01**	**93.59**	**93.01**	**92.91**	**91.54**	92.33	**87.71**
ReLU	89.85	91.77	90.22	90.52	91.59	91.41	91.59	89.56	91.12	86.85
Leaky ReLU (α = 0.01)	89.70	91.85	90.54	90.77	91.92	90.77	91.82	89.42	91.28	86.75
ELU	89.07	91.52	90.61	90.67	91.39	90.27	91.27	89.91	91.01	86.35
Swish	89.81	92.12	90.75	91.09	92.42	91.89	91.84	90.41	91.63	87.21
Softplus	89.35	91.24	89.53	90.22	91.02	91.31	91.41	89.40	91.31	83.41
Mish	89.93	92.48	90.67	91.52	92.62	91.99	92.01	90.67	**92.45**	87.29
GELU	89.88	92.21	90.71	90.87	92.77	91.88	91.77	90.11	91.60	87.18

Table 8.4: Experimental results on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported.

Activation Function	GN	ResNeXt-50	Stochastic Depth	Shufflenet V2	DLA	RegNet	NASNet	RIR	Xception	Le-Net	EN-B0
TanhSoft-1	**92.96**	**93.12**	**92.97**	90.42	**91.18**	**92.67**	**93.18**	**92.55**	90.98	**69.85**	**87.52**
TanhSoft-2	**92.87**	**92.97**	**93.01**	90.27	**91.07**	**92.61**	**93.02**	**92.65**	90.92	**69.77**	**87.71**
TanhSoft-3	**92.77**	**92.82**	**92.85**	89.91	**91.12**	**92.77**	**93.12**	**92.47**	90.67	**69.95**	**87.31**
ReLU	91.07	91.52	91.14	88.54	89.47	90.25	91.09	90.65	90.05	67.05	85.96
Leaky ReLU (α = 0.01)	90.97	91.37	91.22	88.49	89.59	90.39	91.37	90.45	90.09	67.14	86.17
ELU	91.28	91.72	91.49	88.70	89.23	90.47	91.01	90.55	90.45	67.89	85.72
Swish	91.45	91.92	91.59	90.09	89.77	90.41	91.89	90.09	91.12	67.98	86.42
Softplus	90.42	91.32	91.01	88.12	89.63	90.32	91.21	90.52	89.87	66.62	85.11
Mish	91.72	91.79	91.57	**90.55**	90.31	90.67	92.09	91.06	**91.22**	68.03	86.42
GELU	90.65	91.71	91.77	89.01	90.12	90.62	92.27	90.77	91.01	67.66	86.27

Table 8.5: Experimental results on CIFAR10 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported.

Tiny Imagenet

The ImageNet Large Scale Visual Recognition Challenge(ILSVRC) is one of the most popular benchmarks for image classification problems. Tiny ImageNet Challenge is a

Activation Function	VGG-16	WRN 28-10	ResNet-34	PA-ResNet-34	DN-121	DN-169	IN-V3	MN-V2	SN	SQ-Net
TanhSoft-1	**63.11**	**69.70**	**65.11**	**64.54**	**69.12**	68.07	**70.89**	**66.22**	**65.20**	**61.09**
TanhSoft-2	**62.78**	**69.60**	**65.65**	**64.77**	**69.37**	68.02	**70.58**	**66.52**	**65.01**	**61.22**
TanhSoft-3	**62.41**	**69.59**	**65.65**	**64.45**	**69.01**	67.99	**70.67**	**65.81**	**65.01**	**60.98**
ReLU	57.03	67.57	63.52	61.52	67.11	66.22	69.25	63.89	63.52	60.27
Leaky ReLU ($\alpha = 0.01$)	57.17	68.05	63.12	61.82	66.98	67.15	69.01	64.01	63.64	60.29
ELU	56.24	67.49	63.89	61.41	67.34	66.89	69.39	63.52	63.59	60.71
Swish	60.14	68.54	64.57	63.87	68.01	66.89	69.67	64.76	64.85	60.75
SoftPlus	54.22	66.84	61.93	62.22	66.99	66.82	68.91	63.22	62.56	59.98
Mish	60.11	69.09	64.22	63.87	68.52	**68.19**	69.77	65.18	64.89	60.47
GELU	59.77	68.86	64.47	63.69	67.89	67.15	69.68	64.59	64.72	60.09

Table 8.6: Experimental results on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported.

Activation Function	GN	ResNeXt-50	Stochastic Depth	Shufflenet V2	DLA	RegNet	NASNet	RIR	Xception	Le-Net	EN-B0
TanhSoft-1	**71.22**	**71.17**	**69.56**	**63.01**	**64.87**	**68.87**	**71.05**	**65.12**	66.12	**36.82**	**54.52**
TanhSoft-2	**71.42**	**71.02**	**69.77**	**63.07**	**65.01**	**68.52**	**71.22**	**64.99**	66.36	**36.79**	**54.44**
TanhSoft-3	**71.01**	**71.39**	**69.22**	62.70	**64.62**	**68.77**	**70.87**	**65.20**	65.76	**37.09**	**54.18**
ReLU	70.09	69.22	67.97	61.22	63.06	66.02	68.56	63.22	65.32	32.26	52.87
Leaky ReLU ($\alpha = 0.01$)	69.87	69.55	67.75	61.42	63.45	65.87	68.99	63.59	63.17	32.89	53.11
ELU	70.22	69.29	67.55	61.21	63.15	65.99	68.42	63.67	63.34	33.99	52.78
Swish	70.27	69.42	68.22	61.67	63.39	66.99	69.22	63.85	**66.72**	34.99	53.38
Softplus	69.97	68.86	67.55	61.29	62.78	65.52	68.64	62.89	64.89	32.79	52.51
Mish	70.62	69.89	68.19	62.77	63.49	67.22	69.10	63.87	66.46	35.17	53.79
GELU	70.67	69.58	68.11	62.52	63.77	67.24	68.35	63.72	65.81	34.99	53.45

Table 8.7: Experimental results on CIFAR100 dataset. Top-1 accuracy(in %) for mean of 10 different runs have been reported.

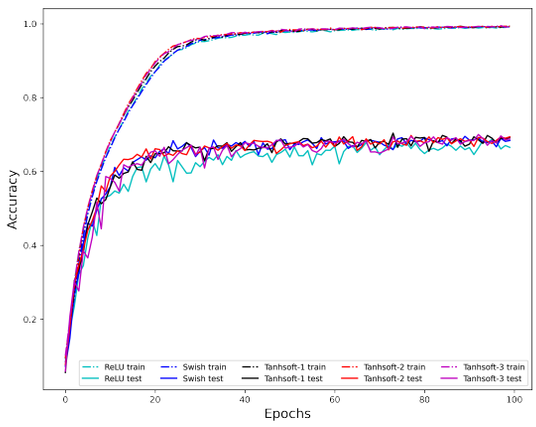

Figure 8.9: Top-1 Train and Test accuracy (higher is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

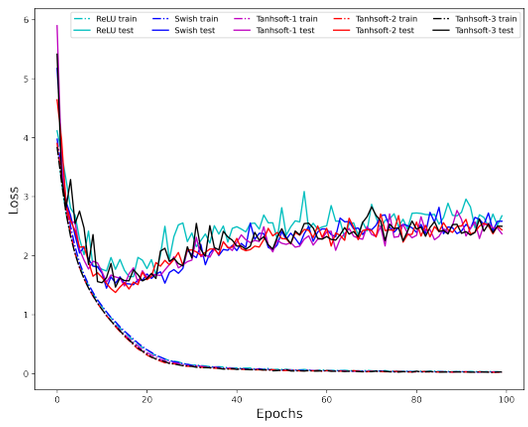

Figure 8.10: Top-1 Train and Test loss (lower is better) on CIFAR100 dataset with WideResNet 28-10 model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

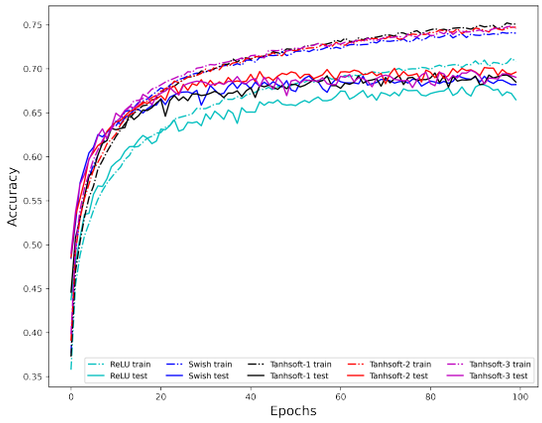

Figure 8.11: Top-1 Train and Test accuracy (higher is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

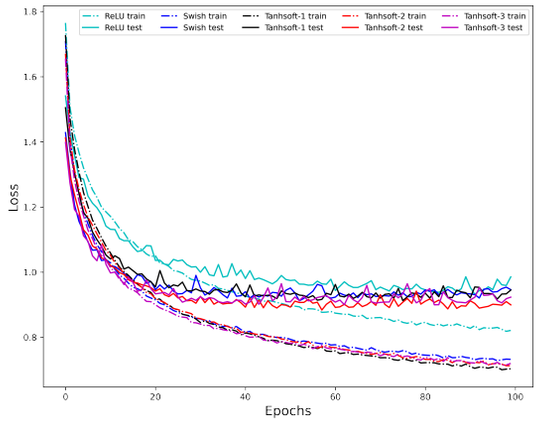

Figure 8.12: Top-1 Train and Test loss (lower is better) on CIFAR10 dataset with LeNet model for ReLU, Swish, TanhSoft-1, TanhSoft-2, and TanhSoft-3.

similar type of challenges like ILSVRC for image classification, which has a smaller dataset and fewer image classes. The database has images of size 64×64 with 200 image classes with a training dataset of 100,000 images, a validation dataset of 10,000 images, and a test dataset of 10,000 images. Each class has 500 training images, 50 validation images, and 50 test images. We have reported results for top-1 accuracy for mean of 5 runs in Table 8.8 on WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)) model. The network is trained with He Normal initializer (He *et al.* (2015*b*)), a batch size of 32, Adam optimizer (Kingma and Ba (2015)), 0.2 dropout rate (Srivastava *et al.* (2014)), initial learning rate(lr rate) 0.01, and reduce lr rate by a factor of 10 after every 50 epochs up-to 250 epochs. We have used data augmentation method in this database.

8.5.2 Object Detection

Object Detection is an important problem in computer vision. We have considered the Pascal VOC dataset (Everingham *et al.* (2010)) for our experiments. Results are reported on Single Shot MultiBox Detector(SSD) 300 model (Liu *et al.* (2016)), and VGG-16(with batch-normalization) is considered as the backbone network. The model is trained on Pascal VOC 07+12 training data, and model performance is evaluated on Pascal VOC 2007 test data. The model has been trained with a batch size of 8,

Activation Function	Wide ResNet 28-10 Model
TanhSoft-1	**62.01**
TanhSoft-2	**62.28**
TanhSoft-3	**62.17**
ReLU	60.35
Swish	60.69
Leaky ReLU(α = 0.01)	60.62
ELU	60.02
Softplus	59.81
Mish	60.77
GELU	60.72

Table 8.8: Experimental results on Tiny ImageNet dataset. Mean of 5 different runs for top-1 accuracy(in $\%$) have been reported.

$5e^{-4}$ weight decay for 120000 iterations, 0.001 learning rate, SGD optimizer (Robbins and Monro (1951); Kiefer and Wolfowitz (1952)) with 0.9 momentum. No pre-trained weight is used in the network. The mean average precision(mAP) is reported in Table 8.9 for a mean of 5 different runs.

Activation Function	mAP
TanhSoft-1	**77.9**
TanhSoft-2	**78.0**
TanhSoft-3	**77.8**
ReLU	77.2
Swish	77.3
Leaky ReLU(α = 0.01)	77.2
ELU	75.1
Softplus	74.2
Mish	77.5
GELU	77.3

Table 8.9: Object Detection results on SSD 300 model in Pascal-VOC dataset .

8.5.3 Semantic Segmentation

Semantic segmentation is a very important problem in computer vision. We have shown our experimental results on the CityScapes dataset (Cordts *et al.* (2016)). CityScapes training data with U-net model (Ronneberger *et al.* (2015)) is trained for 250 epochs, with adam optimizer (Kingma and Ba (2015)), with batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)), and learning rate $5e^{-3}$. Pixel Accuracy and mean Intersection-Over-Union (mIOU) on test data have been reported on Table 8.10 for mean of 5 different runs.

Activation Function	Pixel Accuracy	mIOU
TanhSoft-1	**80.71**	**70.45**
TanhSoft-2	**80.62**	**70.34**
TanhSoft-3	**80.65**	**70.37**
ReLU	79.54	69.39
Swish	79.87	69.68
Leaky ReLU(α = 0.01)	79.59	69.48
ELU	79.12	68.12
Softplus	78.89	68.04
Mish	80.39	69.87
GELU	79.69	69.62

Table 8.10: semantic segmentation results on U-NET model in CityScape dataset.

8.5.4 Machine Translation

Machine Translation is a deep learning technique to translate from one language to another. For this problem, WMT 2014 English→German dataset is used. It has 4.5 million training sentences, and model performance is evaluated on the newstest2014 dataset using the BLEU score metric. We have consider an Attention-based multi-head transformer model (Vaswani *et al.* (2017)) for our experiments. A 8-head transformer model is considered with 0.1 dropout (Srivastava *et al.* (2014)), Adam optimizer (Kingma and Ba (2015)), and trained for 100000 steps. Other hyper-parameters are tried to retain similar as mentioned in the original paper (Vaswani *et al.* (2017)). We have reported a Mean of 5 runs has on Table 8.11 on the test dataset(newstest2014).

Activation Function	BLEU Score on the newstest2014 dataset
TanhSoft-1	**26.7**
TanhSoft-2	**26.7**
TanhSoft-3	**26.6**
ReLU	26.2
Swish	26.4
Leaky ReLU(α = 0.01)	26.3
ELU	25.1
Softplus	23.6
Mish	26.3
GELU	26.2

Table 8.11: Machine translation results on transformer model in WMT-2014 dataset.

8.6 Comparison With Baselines

Based on all the experiments given in earlier sections, we observe that TanhSoft-1, TanhSoft-2, and TanhSoft-3, beats or performs equally well in most cases when compared with the baseline activation functions and under-performs marginally on rare occasions, and we provide a detailed comparison of the proposed activations with the baseline activations in Table 8.12. Table 8.12 contains the total number of cases in which the proposed activations performs better, equal or less than the baseline activations. The proposed activations outperform concerning model performance in all cases compared to ReLU, Leaky ReLU, ELU, and Softplus. Also, compared to Swish, Mish, and GELU, the proposed activations outperform most cases while under-performing on infrequent occasions.

Baselines	ReLU	Leaky ReLU	ELU	Swish	Softplus	GELU	Mish
TanhSoft-1 $>$ Baseline	49	49	49	47	49	48	44
TanhSoft-1 $=$ Baseline	0	0	0	0	0	0	0
TanhSoft-1 $<$ Baseline	0	0	0	2	0	1	5
TanhSoft-2 $>$ Baseline	49	49	49	47	49	48	44
TanhSoft-2 $=$ Baseline	0	0	0	0	0	0	0
TanhSoft-2 $<$ Baseline	0	0	0	2	0	1	5
TanhSoft-3 $>$ Baseline	49	49	49	46	49	47	43
TanhSoft-3 $=$ Baseline	0	0	0	0	0	0	0
TanhSoft-3 $<$ Baseline	0	0	0	3	0	2	6

Table 8.12: Baseline table for TanhSoft-1, TanhSoft-2, and TanhSoft-3 based on all the experiments. The numbers represents the total number of models in which TanhSoft-1, TanhSoft-2, and TanhSoft-3 outperforms, equal or underperforms when compared to baseline activation functions

8.7 Computational Time Comparison

We have reported the computational time comparison for TanhSoft-1, TanhSoft-2, and TanhSoft-3 and the baseline activation functions for both forward and backward pass on a 32×32 RGB image in ResNet-34 model in Table 8.13 for the mean of 100 runs. All the runs are performed on an NVIDIA Tesla V100 GPU with 16GB ram. The computational time for both forward and backward passes are reported in milliseconds (μs). From Table 8.13, we notice that due to the nonliterary of the proposed activations, the computational time for both forward and backward pass is slightly higher than ReLU (in milliseconds) while it is similar to Mish and better than GELU. Also, due to the non-linearity of the proposed activations, there is a trade-off between state-of-the-art model performances and computational time. From the experimental section, we notice that compared to ReLU networks with the proposed activations networks, the model performance has increased significantly, but the computational time increased marginally.

8.8 Conclusion

We have explored three novel trainable activation functions in this work, TanhSoft-1, TanhSoft-2, and TanhSoft-3. The proposed functions are zero-centred, non-monotonic, non-zero negative bounded curve, continuous, and differentiable. In the beginning, we

Activation Function	Forward Pass(STD)	Backward Pass(STD)
TanhSoft-1	15.26($\pm$1.93) μs	13.40($\pm$2.50) μs
TanhSoft-2	15.68($\pm$2.07) μs	11.55($\pm$1.80) μs
TanhSoft-3	15.34($\pm$2.16) μs	12.47($\pm$1.93) μs
ReLU	12.38($\pm$1.10) μs	9.51($\pm$1.24) μs
Swish	13.21($\pm$1.57) μs	10.91($\pm$1.91) μs
Leaky ReLU (α = 0.01)	12.42($\pm$2.05) μs	10.22($\pm$1.58) μs
ELU	12.56($\pm$2.12) μs	10.27($\pm$1.88) μs
Softplus	12.50($\pm$2.35) μs	10.41($\pm$2.37) μs
Mish	14.21($\pm$2.55) μs	14.60($\pm$3.45) μs
GELU	19.02($\pm$2.90) μs	19.88($\pm$3.41) μs

Table 8.13: Runtime comparison for the forward and backward passes for TanhSoft-1, TanhSoft-2, and TanhSoft-3 and baseline activation functions for a 32$\times$ 32 RGB image in ResNet-34 model.

have conducted experiments with the three activations with constant hyper-parameters, and we found that these activations perform equally or slightly better than ReLU. Later, we tune the hyper-parameters via backpropagation and make the proposed activations trainable. In this case, we found a considerable change in results (Top-1 accuracy or mAP or mIOU or BLEU score), and they perform far better than ReLU or the other baseline activations in most of the experiments. It shows that introducing a trainable parameter plays an essential role in activation functions, and a non-zero bounded negative part & trainable parameters result in better performance. We have used hyperparameters and models with the ReLU activation function and then replace ReLU with other baseline activations & the proposed activations to compare model performances. Our empirical evaluation on different deep learning tasks like Image classification, Object Detection, Semantic Segmentation, Machine Translation in a variety of complex models on datasets like MNIST, Fashion MNIST, SVHN, CIFAR10, CIFAR100, Tiny ImageNet, Pascal VOC, CityScapes, and WMT 2014 shows that the proposed activation functions produce state-of-the-art results and have an excellent potential to replace the widely used activations functions like ReLU, Leaky ReLU, ELU, Softplus, Swish, Mish, and GELU.

CHAPTER 9

EIS [1]

9.1 Introduction

Multi-layered neural networks are widely used to learn nonlinear functions from complex data. An activation function is an integral part of neural networks that provides essential non-linearity. A universal activation function may not be suitable for all datasets, and selecting an appropriate activation function for the task at hand is important. Nevertheless, a piecewise activation function, Rectified Linear Unit (ReLU) (Nair and Hinton (2010)), defined as $\max(x, 0)$, is widely used due to its simplicity, convergence speed, and lesser training time. Despite its simplicity and better convergence rate than Sigmoid and Tanh, ReLU has drawbacks like non-zero mean, negative missing, unbounded output, and dying ReLU, to name a few (see (Zhou *et al.* (2020))). The prominent drawback of ReLU is the dying ReLU that provides zero output for negative input. Many novel activation functions are built to overcome this problem. Many activation functions resolved it by simply defining a piecewise function that resembles ReLU for positive input and taking non-zero values for negative input. Swish (Ramachandran *et al.* (2017)) is proposed by a team of researchers from Google Brain through an exhaustive search (Negrinho and Gordon (2017)) and reinforcement learning techniques (Baker *et al.* (2016)). Experimental evaluation shows that Swish performs better than ReLU on different deep learning problems. Swish is different from such piecewise activation functions in the sense that it is a product of two smooth functions and manages to remain close to ReLU for positive input and takes small negative values for the negative input.

[1] This chapter is a slightly modified version of the paper accepted at ICANN conference Biswas *et al.* (2021*b*).

9.2 Related works

Several activation functions have been proposed as a substitute for ReLU that can overcome its drawbacks. Because of the dying ReLU problem, it has been observed that a large fraction of neurons become inactive due to zero outcomes. Another issue that activation functions face is that during the flow of gradient in the network, the gradient can become zero or diverge to infinity, which is commonly known as vanishing and exploding gradient problems. Leaky Relu (Maas *et al.* (2013*a*)) has been introduced with a small negative linear component to solve the dying ReLU problem and has shown improvement over ReLU. A hyper-parametric component is incorporated in PReLU (He *et al.* (2015*b*)) to find the best value in the negative linear component. Many other improvements have been proposed over the years - Leaky ReLU (Maas *et al.* (2013*a*)), Randomized Leaky Rectified Linear Units (RReLU) (Xu *et al.* (2015*a*)), Exponential Linear Unit (ELU) (Clevert *et al.* (2016)), Inverse Square Root Linear Units (ISRLUs) (Carlile *et al.* (2017)), GELU (Hendrycks and Gimpel (2020)), Swish (Ramachandran *et al.* (2017)), and Parametric Rectified Linear Unit (PReLU) (He *et al.* (2015*b*)) to name a few. However, none of the above-mentioned activation functions has come close to ReLU in terms of popularity. Recently, Swish (Ramachandran *et al.* (2017)) has been proposed. Swish is a one-parameter family of activation functions defined as $x\,\mathrm{sigmoid}(\beta x)$ and managed to gain attention from the deep learning community. Some other hyper-parametrized families of activation functions include Soft-Root-Sign (Zhou *et al.* (2020)) activation function.

9.3 Research Contribution

In this chapter, we have proposed three parametric smooth activation functions and shown that they outperform widely used activation functions, including ReLU and Swish. To validate the performance of these activations, we have performed a wide range of experiments on four very important and different deep learning problems like image classification, object detection, semantic segmentation, and machine translation, and the results are reported in the experiment section.

9.4 EIS-1, EIS-2, and EIS-3

We have proposed three families of activation functions with learnable parameters. We call them EIS-1 ($\mathcal{F}_1(x;\alpha,\beta)$), EIS-2 ($\mathcal{F}_2(x;\gamma)$), and EIS-3 ($\mathcal{F}_3(x;\delta,\theta)$). They are defined as follows:-

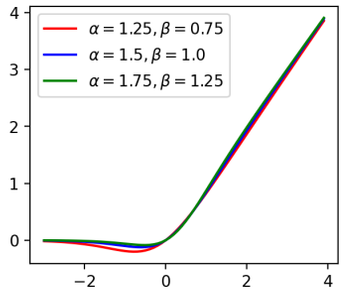

Figure 9.1: Graph of $\mathcal{F}_1(x;\alpha,\beta)$ for different values of α,β.

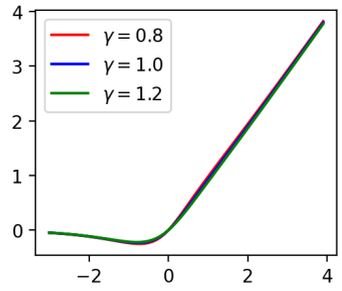

Figure 9.2: Graph of $\mathcal{F}_2(x;\gamma)$ for different values of γ.

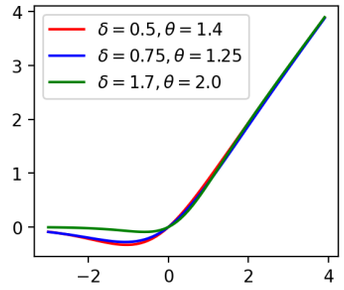

Figure 9.3: Graph of $\mathcal{F}_3(x;\delta,\theta)$ for different values of δ,θ.

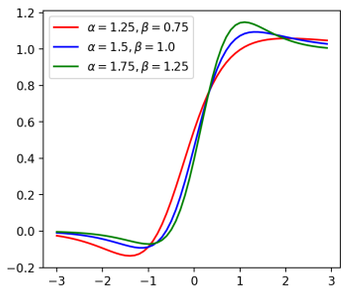

Figure 9.4: Graph of first derivative of $\mathcal{F}_1(x;\alpha,\beta)$ for different values of α,β.

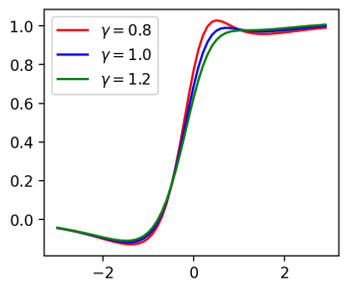

Figure 9.5: Graph of first derivative of $\mathcal{F}_2(x;\gamma)$ for different values of γ.

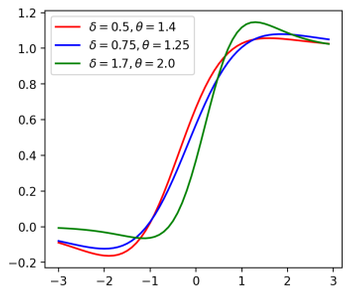

Figure 9.6: Graph of first derivative of $\mathcal{F}_3(x;\delta,\theta)$ for different values of δ,θ.

$$\mathcal{F}_1(x;\alpha,\beta) = \frac{x\ln(1+e^x)}{x+\alpha e^{-\beta x}}, \tag{9.1}$$

$$\mathcal{F}_2(x;\gamma) = \frac{x\ln(1+e^x)}{\sqrt{\gamma+x^2}}, \tag{9.2}$$

$$\mathcal{F}_3(x;\delta,\theta) = \frac{x}{1+\delta e^{-\theta x}}. \tag{9.3}$$

The derivative of the above activations are:-

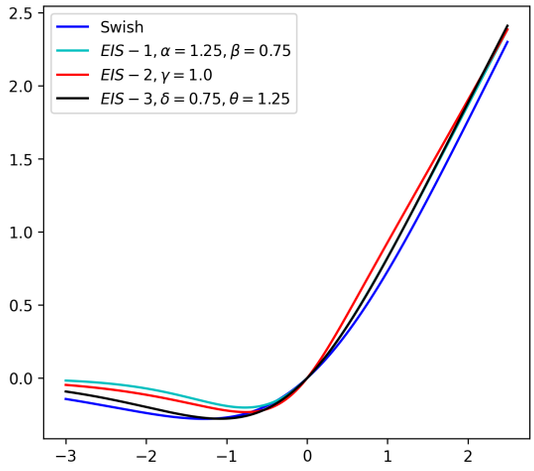

Figure 9.7: Graph of Swish, $\mathcal{F}_1(x;\alpha,\beta)$, $\mathcal{F}_2(x;\gamma)$ and $\mathcal{F}_3(x;\delta,\theta)$

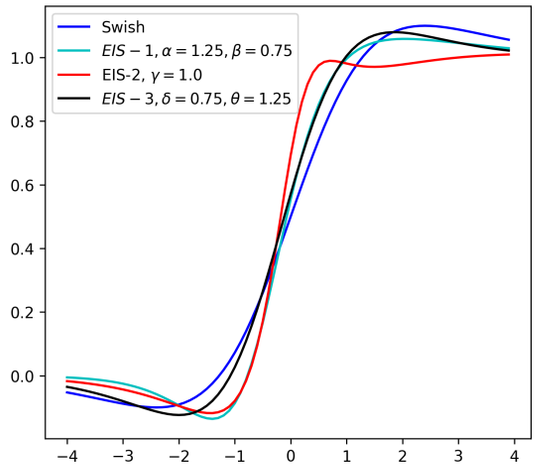

Figure 9.8: Graph of first order derivatives of Swish, $\mathcal{F}_1(x;\alpha,\beta)$, $\mathcal{F}_2(x;\gamma)$, and $\mathcal{F}_3(x;\delta,\theta)$

$$\frac{d}{dx}\mathcal{F}_1(x;\alpha,\beta) = \frac{\ln(1+e^x)}{x+\alpha e^{-\beta x}} + \frac{x}{x+\alpha e^{-\beta x}}\,\frac{e^x}{1+e^x} - \frac{(1-\alpha\beta e^{-\beta x})(x\ln(1+e^x))}{(x+\alpha e^{-\beta x})^2}, \tag{9.4}$$

$$\frac{d}{dx}\mathcal{F}_2(x;\gamma) = \frac{\ln(1+e^x)}{\sqrt{\gamma+x^2}} + \frac{x}{\sqrt{\gamma+x^2}}\,\frac{e^x}{1+e^x} - \frac{x^2\ln(1+e^x)}{(\gamma+x^2)^{\frac{3}{2}}}, \tag{9.5}$$

$$\frac{d}{dx}\mathcal{F}_3(x;\delta,\theta) = \frac{1}{1+\delta e^{-\theta x}} + \frac{\delta\theta x e^{-\theta x}}{(1+\delta e^{-\theta x})^2}. \tag{9.6}$$

The hyper-parameters α,β for EIS-1, γ for EIS-2, and δ,θ for EIS-3 controls the slope of the functions in both negative and positive axes as evident from figure 9.1, 9.2, and 9.3. For square root function, we have considered only the positive branch . Note that $\mathcal{F}_1(x;0,\beta)$ and $\mathcal{F}_2(x;0)$ recovers the Softplus function while $\mathcal{F}_3(x;0,\theta)$ recovers the identity function x. Moreover,

$$\lim_{\delta\to\infty}\mathcal{F}_3(x;\delta,\theta) = 0 \quad \forall x\in\mathbb{R}. \tag{9.7}$$

Graph of some functions from these three families are given in Figures 9.1, 9.2, and 9.3. The first-order derivatives of these functions are shown in Figures 9.4, 9.5, and 9.6. Moreover, one function from each of these three families and their derivatives are compared with Swish in Figures 9.7 and 9.8. As evident from graphs, chosen functions of these three subfamilies have bounded negative domain, smooth derivative and, non-

monotonic curve like Swish.

9.5 Experiments with EIS-1, EIS-2, and EIS-3

In all the experiments, the learnable parameters in EIS-1, EIS-2, and EIS-3 are first initialized and then updated using the back propagation (LeCun *et al.* (1989)) algorithm (see He *et al.* (2015*b*)). For a single layer, the gradient of a hyper-parameter η is:

$$\frac{\partial E}{\partial \eta} = \sum_{x} \frac{\partial E}{\partial F(x)} \frac{\partial F(x)}{\partial \eta} \tag{9.8}$$

where E is the objective function, $\eta \in \{\alpha, \beta, \gamma, \delta, \theta\}$ and $F(x) \in \{\mathcal{F}_1(x; \alpha, \beta), \mathcal{F}_2(x; \gamma), \mathcal{F}_3(x; \delta, \theta)\}$. Table 9.1 provides a detailed comparison of EIS-1, EIS-2, and EIS-3 with seven baseline activation functions, ReLU (Nair and Hinton (2010)), Leaky Relu (Maas *et al.* (2013*a*)), ELU (Clevert *et al.* (2016)), Softplus (Zheng *et al.* (2015)), Swish (Ramachandran *et al.* (2017)), Mish (Misra (2020)), and GELU (Hendrycks and Gimpel (2020)). We have given detailed experimental setup and results for different deep learning problems like image classification, object detection, semantic segmentation, and Machine translation in the next section. We have initialized the learnable parameters at $\alpha = 1.25,\ \beta = 0.75$ for EIS-1, $\gamma = 1.0$ for EIS-2, and $\delta = 0.75,\ \theta = 1.25$ for EIS-3 throughout all the experiments and they are updated in network models during back-propagation.

Baselines	ReLU	Leaky ReLU	ELU	Swish	Softplus	Mish	GELU
EIS-1 > Baseline	29	29	29	28	29	26	29
EIS-1 = Baseline	0	0	0	0	0	0	0
EIS-1 < Baseline	0	0	0	1	0	3	0
EIS-2 > Baseline	29	29	29	28	29	26	29
EIS-2 = Baseline	0	0	0	0	0	0	0
EIS-2 < Baseline	0	0	0	1	0	3	0
EIS-3 > Baseline	29	29	29	28	29	26	29
EIS-3 = Baseline	0	0	0	0	0	0	0
EIS-3 < Baseline	0	0	0	1	0	3	0

Table 9.1: Baseline table for EIS-1, EIS-2, and EIS-3. The integers represents the total number of models in which EIS-1, EIS-2, and EIS-3 outperforms, equal or underperforms when compared to baseline activations

It is evident from the baseline table 9.1 that EIS-1, EIS-2, and EIS-3 outperform when compared to baseline activations in most cases and perform equally or under-

perform occasionally. The forward pass is implemented in both Pytorch (Paszke *et al.* (2019)) & Tensorflow-Keras (Chollet *et al.* (2015)) API and automatic differentiation updates the parameters. All the experiments are conducted on an NVIDIA tesla V-100 GPU with 16GB RAM.

9.5.1 Image Classification:

We have reported results for image classification with six benchmarking databases like MNIST, Fashion MNIST, Street View House Numbers (SVHN), CIFAR10, CIFAR100, and Tiny Imagenet. A brief description of the databases and experimental setup is as follows.

MNIST:

MNIST (LeCun *et al.* (2010)) is a well established standard databases consisting of 28 $\times$ 28 pixels grey-scale images of handwritten digits from 0 to 9. The dataset consists of 60k training images and 10k testing 28×28 grey-scale images. We consider a custom 8-layer homogeneous convolutional neural network (CNN) architecture to carried out experiments on MNIST. Channel depths of size 128 (twice), 64 (thrice), 32 (twice), a dense layer of size 128, Max-pooling layer(thrice), batch-normalization (Ioffe and Szegedy (2015)) and dropout (Srivastava *et al.* (2014)) is being used on the CNN architecture. No data augmentation is used. The results are reported in Table 9.2.

Fashion-MNIST:-

Fashion-MNIST (Xiao *et al.* (2017)) is a database consisting of 28 $\times$ 28 pixels grey-scale images of Zalando's ten fashion items class like T-shirt, Trouser, Coat, Bag, etc. It's consists of 60k training examples and 10k testing examples. No data augmentation is used. The same CNN model architecture used in the MNIST dataset is also used for this database as well for training and testing purpose and, the results are given in Table 9.2.

Activation Function	5-fold mean Accuracy on MNIST data	5-fold mean Accuracy on Fashion MNIST data	5-fold mean Accuracy on SVHN data
EIS-1	**99.39**	**93.32**	**95.46**
EIS-2	**99.38**	**93.29**	**95.45**
EIS-3	**99.44**	**93.30**	**95.43**
ReLU	99.17	92.95	95.20
Swish	99.21	92.92	95.21
Leaky ReLU	99.18	92.99	95.18
ELU	99.15	92.83	95.10
Softplus	99.02	92.51	95.01
GELU	99.20	93.08	95.23
Mish	99.26	93.16	95.29

Table 9.2: Results on MNIST, Fashion-MNIST and SVHN Datasets.

Street View House Numbers (SVHN) Database:

SVHN (Netzer *et al.* (2011)) is a popular computer vision database consists of real-world house numbers with 32×32 RGB images. The database has 73257 training images and 26032 testing images. The database has a total of 10 classes. We have used the data augmentation method in this database. The same CNN model architecture used in the MNIST dataset is also used for this database as well for training and testing purpose and, the results are given in Table 9.2.

CIFAR:

The CIFAR (Krizhevsky (2009)), is another standard well established computer-vision dataset that is generally used to establish the efficacy of deep learning models. It contains 60k color images of size 32×32, out of which 50k are training images, and 10k are testing images. It has two versions CIFAR 10 and CIFAR100, which contains 10 and 100 target classes, respectively. Top-1 accuracy for mean of 9 different runs is reported on CIFAR10 and CIFAR100 datasets in Table 9.3 and Table 9.4 respectively on ResNet-50 (RN 50) (He *et al.* (2015*a*)), ResNet V2 34 (RN-V2 34) (He *et al.* (2016)), VGG-16 (with Batch-normalization) (Simonyan and Zisserman (2015)), Densenet-121 (DN 121) (Huang *et al.* (2016*a*)), DenseNet-169 (DN 169) (Huang *et al.* (2016*a*)), InceptionNet V3 (IN V3) (Szegedy *et al.* (2015*a*)), SimpleNet (SN) (Hasanpour *et al.* (2016)), MobileNet V2 (MN V2) (Sandler *et al.* (2019)), WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)), ShuffleNet V2 (SF Net) (Ma *et al.* (2018)) and SqueezeNet (SQ Net) (Iandola *et al.* (2016)) models. The networks have been

trained with batch size 128, Adam optimizer Kingma and Ba (2015) with 0.001 learning rate and up-to 100 epochs for all the models mentioned above except SimpleNet and VGG-16 which is trained till 200 epochs. Data augmentation is used for both datasets. Accuracy and loss graphs on WRN 28-10 model with CIFAR100 dataset for ReLU, Swish, EIS-1, EIS-2, and EIS-3 are given in Figures 9.9 and 9.10.

AF	VGG 16	WRN 28-10	RN 50	RN-V2 34	DN 121	DN 169	IN V3	MN V2	SN	SQ Net	SF Net
EIS-1	**90.83**	**92.75**	**91.37**	**91.92**	91.29	91.17	**92.11**	**91.22**	**92.45**	**87.09**	**90.17**
EIS-2	**90.71**	**92.69**	**91.35**	**91.79**	91.17	91.33	**92.02**	**91.11**	**92.37**	**86.99**	**90.02**
EIS-3	**90.79**	**92.81**	**91.30**	**91.97**	91.29	91.31	**92.15**	**91.32**	**92.47**	**87.22**	**90.09**
ReLU	89.62	91.65	90.35	90.52	90.31	90.47	91.25	89.77	91.01	86.72	88.42
Leaky ReLU	89.64	91.77	90.53	90.62	90.69	90.52	91.52	89.71	91.15	86.22	88.40
ELU	89.01	91.22	90.22	90.27	90.23	90.27	91.02	89.09	90.89	86.31	88.31
Swish	89.86	92.01	90.77	90.87	90.71	**91.34**	91.32	90.12	91.41	86.41	89.01
Softplus	89.22	91.36	89.67	89.98	90.12	90.17	91.11	88.99	91.23	85.61	88.01
Mish	90.01	92.23	90.99	90.87	**91.45**	90.77	91.52	90.42	91.99	86.71	89.00
GELU	89.72	92.11	90.78	90.91	90.42	90.73	91.77	90.01	91.52	86.80	89.19

Table 9.3: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on image classification problem on CIFAR10 dataset based on top-1 test accuracy. Top-1 accuracy(in $\%$) for mean of 9 different runs have been reported.

AF	VGG 16	WRN 28-10	RN 50	RN-V2 34	DN 121	DN 169	IN V3	MN V2	SN	SQ Net	SF Net
EIS-1	**62.52**	**69.22**	**65.62**	**65.44**	67.05	64.92	**69.29**	**65.87**	**65.11**	**61.42**	**63.42**
EIS-2	**62.49**	**69.11**	**65.52**	**65.21**	67.01	64.94	**69.27**	**65.71**	**64.99**	**61.23**	**63.23**
EIS-3	**63.01**	**69.21**	**65.61**	**65.49**	67.11	65.19	**69.52**	**65.90**	**65.40**	**61.50**	**63.62**
ReLU	57.25	67.20	64.45	59.89	66.11	64.01	68.11	63.24	63.12	60.12	61.12
Leaky ReLU	57.29	67.86	64.15	60.22	66.82	64.49	68.01	63.27	63.64	60.01	61.03
ELU	56.12	67.58	64.11	59.87	66.11	64.02	67.99	63.02	63.45	60.00	61.07
Swish	60.25	68.22	65.01	60.89	66.92	64.52	68.42	64.11	64.74	60.45	61.15
SoftPlus	54.13	67.01	62.20	59.11	66.20	64.54	68.02	62.98	62.81	59.79	60.89
Mish	60.02	68.99	65.11	62.33	**67.42**	**65.20**	68.51	64.82	64.68	60.12	61.48
GELU	59.89	68.71	64.92	62.45	66.52	64.54	68.40	64.10	64.49	60.03	61.55

Table 9.4: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on image classification problem on CIFAR100 dataset based on top-1 test accuracy. Top-1 accuracy(in $\%$) for mean of 9 different runs have been reported.

Tiny Imagenet

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is the standard and most popular benchmark for image classification problems. The database contains images of size 64×64 with 200 image classes with a training dataset of 100,000 images, a validation dataset of 10,000 images, and a test dataset of 10,000 images. Top-1

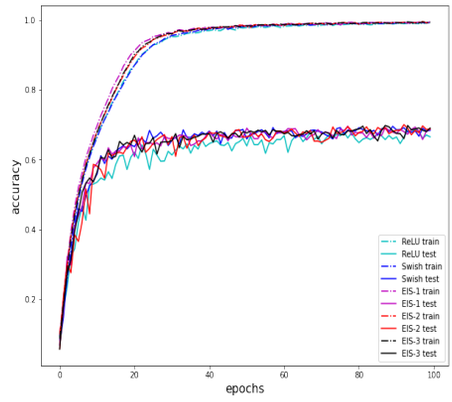

Figure 9.9: Graph for train and test accuracy on CIFAR100 dataset on WideResNet 28-10 model

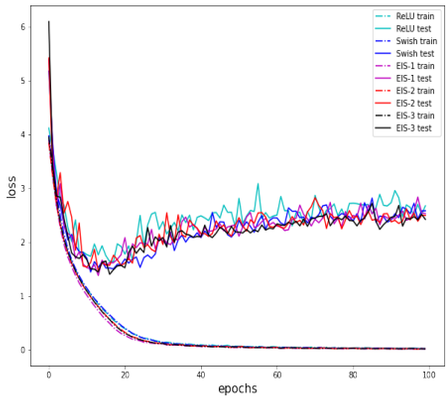

Figure 9.10: Graph for train and test loss on CIFAR100 dataset on WideResNet 28-10 model

accuracy for mean of 5 runs for different activation functions are reported in table 9.5 on WideResNet 28-10 (WRN 28-10) (Zagoruyko and Komodakis (2016)) model. The model is trained with a batch size of 32, He Normal initializer (He *et al.* (2015*b*)), 0.2 dropout rate (Srivastava *et al.* (2014)), adam optimizer, initial learning rate(lr rate) 0.01, and reduce lr rate by a factor of 10 after every 50 epochs up-to 250 epochs. Data augmentation is used.

9.5.2 Object Detection

Object Detection is one of the most important problems in computer vision. We have shown our experimental results on the Pascal VOC dataset (Everingham *et al.* (2010)). Results are reported on Single Shot MultiBox Detector(SSD) 300 model. VGG-16(with batch-normalization) is used as the base network. No pre-trained weight is used in the network. The network is trained on Pascal VOC 07+12 training data and tested model performance on Pascal VOC 2007 test data. The model is trained with a batch size of 8, 0.001 learning rate, SGD optimizer with 0.9 momentum, $5e^{-4}$ weight decay for 120000 iterations. A mean of 5 different runs for the mean average precision(mAP) is reported in table 9.6.

Activation Function	Wide ResNet 28-10 Model
EIS-1	**61.85**
EIS-2	**61.70**
EIS-3	**61.95**
ReLU	60.11
Leaky ReLU	60.05
Swish	60.45
ELU	59.87
Softplus	59.55
Mish	60.61
GELU	60.59

Table 9.5: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Tiny ImageNet dataset on WRN 28-10 model. Results are reported for mean of 5 different runs.

Activation Function	mAP
EIS-1	**77.7**
EIS-2	**77.6**
EIS-3	**77.7**
ReLU	77.2
Swish	77.3
Leaky ReLU	77.2
ELU	75.1
Softplus	74.2
Mish	77.4
GELU	77.3

Table 9.6: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Object Detection problem on SSD 300 model on Pascal-VOC dataset. Results are reported for mean of 5 different runs.

9.5.3 Semantic Segmentation

We carry out our experiment for semantic segmentation task on the Cityscapes dataset (Cordts *et al.* (2016)). We use U-net (Ronneberger *et al.* (2015)) as the base network and train till 250 epochs, with adam optimizer (Kingma and Ba (2015)), learning rate $5e^{-3}$, batch size 32 and Xavier Uniform initializer (Glorot and Bengio (2010)). Mean of 5 different runs for Pixel Accuracy and mean Intersection-Over-Union (mIOU) on test data is reported on table 9.7.

9.5.4 Machine Translation

In this section, we report results for the machine translation problem. For this problem, we use WMT 2014 English→German dataset, which has 4.5 million training sentences, and evaluate model performance on the newstest2014 dataset using BLEU score metric.

We use an Attention-based multi-head transformer model (Vaswani *et al.* (2017)). 8-head attention model is used with Adam optimizer, 0.1 dropout, and trained for 100000 steps. We try to kept other hyper-parameters similar as mentioned in the original paper (Vaswani *et al.* (2017)). Table 9.8 shows the results on the test dataset(newstest2014). A mean of 5 different runs is reported on table 9.8.

Activation Function	Pixel Accuracy	mIOU
EIS-1	**80.55**	**70.34**
EIS-2	**80.61**	**70.29**
EIS-3	**80.51**	**70.27**
ReLU	79.64	69.45
Swish	79.94	69.73
Leaky ReLU	79.71	69.65
ELU	79.05	68.07
Softplus	78.98	68.02
Mish	80.03	69.55
GELU	79.77	69.67

Table 9.7: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on semantic segmentation problem on U-NET model on Cityscapes dataset. Results are reported for mean of 5 different runs.

Activation Function	BLEU Score on the newstest2014 dataset
EIS-1	**26.6**
EIS-2	**26.5**
EIS-3	**26.6**
ReLU	26.2
Swish	26.4
Leaky ReLU	26.3
ELU	25.1
Softplus	23.6
Mish	26.3
GELU	26.2

Table 9.8: Comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 on Machine translation problem on multi-head transformer model on WMT-2014 dataset. Results are reported for mean of 5 different runs.

9.6 Computational Time Comparison

In this section, computational time comparison are reported for baseline activation functions and EIS-1, EIS-2, & EIS-3 for both forward and backward pass for a 32×32 RGB image on the VGG-16 model. All the runs are performed on an NVIDIA Tesla V100 GPU with 16GB ram, and results are reported in Table 9.9 for the mean of 100 runs.

Activation Function	Forward Pass(STD)	Backward Pass(STD)
EIS-1	6.52($\pm$0.99) μs	7.11($\pm$0.96) μs
EIS-2	6.34($\pm$1.17) μs	7.81($\pm$1.74) μs
EIS-3	6.99($\pm$1.01) μs	6.96($\pm$1.34) μs
ReLU	5.10($\pm$1.02) μs	4.95($\pm$0.81) μs
Swish	5.52($\pm$1.11) μs	5.70($\pm$1.05) μs
Leaky ReLU	5.11($\pm$0.59) μs	4.99($\pm$1.01) μs
ELU	5.15($\pm$0.70) μs	5.01($\pm$0.45) μs
Softplus	5.12($\pm$1.01) μs	5.07($\pm$0.99) μs
Mish	6.29($\pm$1.16) μs	5.52($\pm$0.79) μs
GELU	7.59($\pm$1.04) μs	7.89($\pm$1.11) μs

Table 9.9: Runtime comparison between baseline activation functions and EIS-1, EIS-2, & EIS-3 for the forward and backward passes for a 32$\times$ 32 RGB image on VGG-16 model. Results are reported for mean of 100 runs.

9.7 Conclusion

In this chapter, we proposed three parametric activation functions, which we call EIS-1, EIS-2, and EIS-3, and exhibit that they consistently outperform well-known activation functions such as ReLU and Swish, as evident from the baseline table on several well-known datasets and models.

We also advocate through this article that it is time to move away from simple activation functions and adopt comprehensive search schemes on parametric functions to build models. This allows for building more accurate and dependable models. Another scope of future research is to develop a mathematical understanding of reasons leading to improved accuracy.

CHAPTER 10

Conclusion and Future Research Directions

10.1 Conclusion:

In this chapter, we will summarise the contribution of this thesis work. This dissertation mainly focuses on one of the fundamental problems in deep learning- the construction of new novel and practical activation functions. We proposed two types of activation functions; one type is constructed using mathematical approximation theory, and another type of activation function is constructed with traditional handcrafted methods & neural architecture search. Hand-designated activations are widely used in deep learning. As a part of this thesis work, several activations have been proposed, which have been widely tested on standard benchmarking datasets on different deep learning problems like image classification, object detection, semantic segmentation, and machine translation. The proposed functions have been compared with state-of-the-art activation functions like ReLU, Leaky ReLU, Swish, Mish, PAU, GELU etc. The main contribution to this thesis are:

- **Activation's from approximation of known functions:** This types of activations has been introduced in chapter 4, 3, 6, and 7. We show that these four proposed methods can approximate ReLU, Leaky ReLU, or its variants. Also, it is possible to approximate a more general maximum family with the proposed methods.
- **Handcrafted activation functions:** This types of activation functions function have been introduced in chapter 5, 8, and 9. These three proposals are hand-designated. We also show that a few of these functions are actually an approximation by a smooth function of the ReLU activation function.

In this thesis, I have shown that the proposed activation functions can be effectively used in four important and different deep learning problems- image classification, object detection, semantic segmentation, and machine translation. They perform better than the widely used activation functions in most cases. The proposed activation functions can be used in many other domains of deep learning where the artificial neural

network has applications (biomedical engineering, healthcare, weather prediction, pose estimation, action recognition, face recognition etc.). I have obtained good results in some these domains and will use the proposed activation functions in future applied works.

10.2 Summary of the proposed works

In the following subsections, a mathematical and experimental summary of the proposed activation function has been given.

10.2.1 Mathematical Summary of existing and proposed works

A detailed summary of the mathematical properties of existing and proposed works has been reported in Table 10.1.

Activation Function	Zero-Centered	Non-Monotonic	Non-zero Negative	Continuous	Smooth	Trainable
ReLU	Yes	No	No	Yes	No	No
ReLU6	Yes	No	No	Yes	No	No
Leaky ReLU	Yes	No	Yes	Yes	No	No
Parametric ReLU	Yes	No	Yes	Yes	No	Yes
Swish	Yes	Yes	Yes	Yes	Yes	Yes
ELU	Yes	No	Yes	Yes	Yes	No
Softplus	No	No	No	Yes	Yes	No
Mish	Yes	Yes	Yes	Yes	Yes	No
GELU	Yes	Yes	Yes	Yes	Yes	No
PAU	No	Yes	Yes	Yes	Yes	Yes
SAU	Yes	Yes	Yes	Yes	Yes	Yes
SMU	Yes	Yes	Yes	Yes	Yes	Yes
SMU-1	No	Yes	Yes	Yes	Yes	Yes
ErfAct	Yes	Yes	Yes	Yes	Yes	Yes
Pserf	Yes	Yes	Yes	Yes	Yes	Yes
MAU-1	Yes	Yes	Yes	Yes	Yes	Yes
MAU-2	Yes	Yes	Yes	Yes	Yes	Yes
MAU-3	Yes	Yes	Yes	Yes	Yes	Yes
HP-1	No	Yes	Yes	Yes	Yes	Yes
HP-2	No	Yes	Yes	Yes	Yes	Yes
TanhSoft-1	Yes	Yes	Yes	Yes	Yes	Yes
TanhSoft-2	Yes	Yes	Yes	Yes	Yes	Yes
TanhSoft-3	Yes	Yes	Yes	Yes	Yes	Yes
EIS-1	Yes	Yes	Yes	Yes	Yes	Yes
EIS-2	Yes	Yes	Yes	Yes	Yes	Yes
EIS-3	Yes	Yes	Yes	Yes	Yes	Yes

Table 10.1: The relationship and difference between the proposed Activation functions and previously proposed widely used activation functions.

10.2.2 Experimental Summary of the proposed works

A brief summary of the experimental evaluation of the proposed functions has been given in the following subsections. All the experiments have similar experimental setups as presented in chapter 4 on Image Classification, Object detection, and Machine Translation.

Image Classification

In Table 10.2, Top-1 accuracy has been reported on the CIFAR100 dataset for the mean of 5 different runs for the image classification problem.

Activation Function	Shufflenet V2 2.0x	ResNet-50
SAU	73.02 $\pm$ 0.21	76.77 $\pm$ 0.24
SMU	73.69 $\pm$ 0.20	77.48 $\pm$ 0.22
SMU-1	73.31 $\pm$ 0.22	76.74 $\pm$ 0.23
ErfAct	73.20 $\pm$ 0.21	77.04 $\pm$ 0.21
Pserf	73.01 $\pm$ 0.23	77.01 $\pm$ 0.21
MAU-1	73.32 $\pm$ 0.22	77.02 $\pm$ 0.20
MAU-2	73.21 $\pm$ 0.25	77.10 $\pm$ 0.20
MAU-3	73.10 $\pm$ 0.23	76.91 $\pm$ 0.22
HP-1	73.11 $\pm$ 0.22	76.86 $\pm$ 0.22
HP-2	72.91 $\pm$ 0.22	76.56 $\pm$ 0.23
TanhSoft-1	72.52 $\pm$ 0.24	76.60 $\pm$ 0.20
TanhSoft-2	72.65 $\pm$ 0.22	76.45 $\pm$ 0.24
TanhSoft-3	72.46 $\pm$ 0.24	76.20 $\pm$ 0.21
EIS-1	72.20 $\pm$ 0.26	76.10 $\pm$ 0.25
EIS-2	72.31 $\pm$ 0.24	76.21 $\pm$ 0.24
EIS-3	72.59 $\pm$ 0.23	76.40 $\pm$ 0.23

Table 10.2: Comparison between the proposed activations on the CIFAR100 dataset for image classification problem. We report Top-1 test accuracy (in %) for the mean of 5 different runs. mean±std is reported in the table

Object detection

In Table 10.3, mAP has been reported on the Pascal VOC dataset for the mean of 3 different runs for the object detection problem.

Activation Function	mAP
SAU	77.7 $\pm$ 0.12
SMU	78.1 $\pm$ 0.10
SMU-1	77.8 $\pm$ 0.10
ErfAct	78.1 $\pm$ 0.12
Pserf	78.2 $\pm$ 0.12
MAU-1	78.2 $\pm$ 0.10
MAU-2	78.2 $\pm$ 0.12
MAU-3	77.8 $\pm$ 0.14
HP-1	78.0 $\pm$ 0.14
HP-2	77.9 $\pm$ 0.10
TanhSoft-1	77.8 $\pm$ 0.10
TanhSoft-2	77.6 $\pm$ 0.11
TanhSoft-3	77.7 $\pm$ 0.14
EIS-1	77.6 $\pm$ 0.16
EIS-2	77.5 $\pm$ 0.14
EIS-3	77.5 $\pm$ 0.17

Table 10.3: Comparison between the proposed activations on the Pascal VOC dataset for the object detection problem. We report the mAP for the mean of 3 different runs. mean±std is reported in the table.

Machine Translation

In Table 10.4, BLUE score has been reported on the WMT2014 dataset for the mean of 3 different runs for the machine translation problem.

Activation Function	BLEU Score
SAU	26.7 $\pm$ 0.12
SMU	26.8 $\pm$ 0.10
SMU-1	26.6 $\pm$ 0.11
ErfAct	26.8 $\pm$ 0.11
Pserf	26.7 $\pm$ 0.10
MAU-1	26.7 $\pm$ 0.10
MAU-2	26.7 $\pm$ 0.12
MAU-3	26.5 $\pm$ 0.11
HP-1	26.8 $\pm$ 0.10
HP-2	26.7 $\pm$ 0.12
TanhSoft-1	26.6 $\pm$ 0.14
TanhSoft-2	26.7 $\pm$ 0.15
TanhSoft-3	26.5 $\pm$ 0.16
EIS-1	26.5 $\pm$ 0.18
EIS-2	26.4 $\pm$ 0.18
EIS-3	26.5 $\pm$ 0.17

Table 10.4: Comparison between the proposed activations on the WMT2014 dataset for the machine translation problem. We report the BLEU score for the mean of 3 different runs. mean±std is reported in the table.

Computational Time Comparison

In the following Table, computational time comparisons are reported for both forward and backward passes for a 224 $\times$ 224 RGB image on the ResNet-18 model. All the

runs are performed on an NVIDIA RTX 3090 GPU with 24 GB RAM, and results are reported in Table 10.5 for the mean of 100 runs.

Activation Function	Forward Pass(STD)	Backward Pass(STD)
SAU	8.37 ($\pm$1.23) μs	10.82($\pm$0.80) μs
SMU	5.16($\pm$2.13) μs	6.63($\pm$1.93) μs
SMU-1	5.26($\pm$0.90) μs	5.89($\pm$0.81) μs
ErfAct	5.11($\pm$1.63) μs	6.55($\pm$2.13) μs
Pserf	5.55($\pm$1.82) μs	6.09($\pm$1.50) μs
MAU-1	5.38($\pm$0.91) μs	6.83($\pm$1.00) μs
MAU-2	5.42($\pm$1.22) μs	7.13($\pm$1.49) μs
MAU-3	5.29($\pm$0.88) μs	6.24($\pm$1.18) μs
HP-1	15.77($\pm$2.23) μs	22.62($\pm$2.72) μs
HP-2	17.90($\pm$2.47) μs	24.20($\pm$2.41) μs
TanhSoft-1	5.99($\pm$2.03) μs	6.60($\pm$1.94) μs
TanhSoft-2	5.70($\pm$1.92) μs	6.93($\pm$1.89) μs
TanhSoft-3	5.53($\pm$1.46) μs	6.73($\pm$2.02) μs
EIS-1	4.53($\pm$1.04) μs	6.06($\pm$2.11) μs
EIS-2	5.07($\pm$1.33) μs	6.15($\pm$1.01) μs
EIS-3	5.49($\pm$1.62) μs	6.80($\pm$2.01) μs

Table 10.5: Runtime comparison between the proposed activation functions for the forward and backward passes for a 224$\times$ 224 RGB image on the ResNet-18 model. Results are reported for a mean of 100 runs. Experiments are conducted on an NVIDIA RTX 3090 GPU with 24 GB RAM.

10.3 Future Direction

While the proposed functions have been widely tested on different deep learning problems, still there is theoretical scope to improve these works. Some possible future directions are as follows:

- (Xu and Zhang (2021)) shows why deep networks with ReLU activation function converge. It is a possible interesting direction for theoretical work on why deep network with smooth activation functions converges and why they provide better performance than ReLU networks.
- There is a trade-off between training time and model performance for smooth activation functions (proposed functions along with Swish, Mish, GELU, PAU etc.) compared with ReLU, Leaky ReLU etc. It is a possible exciting research direction to reduce the training time for smooth functions without dropping the model performance.

- Deep neural network models are vulnerable to adversarial attacks (Goodfellow *et al.* (2014)). There are few existing defence techniques to prevent these attacks. It is a possible exciting research direction to make a robust defence system against strong adversarial attacks with the help of smooth activation functions.
- Extending the proposed activation functions for complex-valued neural networks.
- It is a possible research direction to define novel activation functions for graph neural networks.

REFERENCES

1. **Baker, B.**, **O. Gupta**, **N. Naik**, and **R. Raskar** (2016). Designing neural network architectures using reinforcement learning. 114, 129

2. **Beckermann, B.**, **V. Kalyagin**, **A. Matos**, and **F. Wielonsky** (2011). How well does the hermite–padé approximation smooth the gibbs phenomenon? *Mathematics of computation*, **80**(274), 931–958. 100

3. **Biswas, K.**, **S. Banerjee**, and **A. K. Pandey** (2021*a*). Orthogonal-padé activation functions: Trainable activation functions for smooth and faster convergence in deep networks. URL https://arxiv.org/abs/2106.09693. 92

4. **Biswas, K.**, **S. Kumar**, **S. Banerjee**, and **A. K. Pandey**, Eis - efficient and trainable activation functions for better accuracy and performance. *In* **I. Farkaš**, **P. Masulli**, **S. Otte**, and **S. Wermter** (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer International Publishing, Cham, 2021*b*. ISBN 978-3-030-86340-1. URL https://doi.org/10.1007/978-3-030-86340-1_21. ii, 129

5. **Biswas, K.**, **S. Kumar**, **S. Banerjee**, and **A. K. Pandey** (2021*c*). Erfact and pserf: Non-monotonic smooth trainable activation functions. URL https://arxiv.org/abs/2109.04386. ii, 56, 76

6. **Biswas, K.**, **S. Kumar**, **S. Banerjee**, and **A. K. Pandey** (2021*d*). Sau: Smooth activation function using convolution with approximate identities. URL https://arxiv.org/abs/2109.13210. ii, 11

7. **Biswas, K.**, **S. Kumar**, **S. Banerjee**, and **A. K. Pandey** (2021*e*). Tanhsoft—dynamic trainable activation functions for faster learning and better performance. *IEEE Access*, **9**, 120613–120623. URL https://doi.org/10.1109/ACCESS.2021.3105355. ii, 112

8. **Biswas, K.**, **S. Kumar**, **S. Banerjee**, and **A. K. Pandey**, Smooth maximum unit: Smooth activation function for deep networks using smoothing maximum technique. *In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. URL https://doi.org/10.1109/CVPR52688.2022.00087. ii, 32

9. **Bochkovskiy, A.**, **C.-Y. Wang**, and **H.-Y. M. Liao** (2020). Yolov4: Optimal speed and accuracy of object detection. 2, 57

10. **Brown, T. B.**, **B. Mann**, **N. Ryder**, **M. Subbiah**, **J. Kaplan**, **P. Dhariwal**, **A. Neelakantan**, **P. Shyam**, **G. Sastry**, **A. Askell**, **S. Agarwal**, **A. Herbert-Voss**, **G. Krueger**, **T. Henighan**, **R. Child**, **A. Ramesh**, **D. M. Ziegler**, **J. Wu**, **C. Winter**, **C. Hesse**, **M. Chen**, **E. Sigler**, **M. Litwin**, **S. Gray**, **B. Chess**, **J. Clark**, **C. Berner**, **S. McCandlish**, **A. Radford**, **I. Sutskever**, and **D. Amodei** (2020). Language models are few-shot learners. 2, 33, 57

11. **Carlile, B.**, **G. Delamarter**, **P. Kinney**, **A. Marti**, and **B. Whitney** (2017). Improving deep learning by inverse square root linear units (isrlus). 113, 130

12. **Chen, C.-T.** and **W. Chang** (1996). A feedforward neural network with function shape autotuning. *Neural Networks*, **9**, 627–641. 114

13. **Chollet, F.** (2017). Xception: Deep learning with depthwise separable convolutions. 20, 43, 64, 81, 120

14. **Chollet, F.** *et al.* (2015). Keras. `https://keras.io`. 16, 38, 78, 96, 134

15. **Clevert, D.-A.**, **T. Unterthiner**, and **S. Hochreiter** (2016). Fast and accurate deep network learning by exponential linear units (elus). 2, 4, 12, 32, 39, 56, 74, 75, 92, 100, 112, 113, 130, 133

16. **Cordts, M.**, **M. Omran**, **S. Ramos**, **T. Rehfeld**, **M. Enzweiler**, **R. Benenson**, **U. Franke**, **S. Roth**, and **B. Schiele** (2016). The cityscapes dataset for semantic urban scene understanding. 28, 52, 69, 87, 108, 125, 138

17. **Deng, J.**, **W. Dong**, **R. Socher**, **L.-J. Li**, **K. Li**, and **L. Fei-Fei**, Imagenet: A large-scale hierarchical image database. *In 2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009. 60, 67, 85

18. **Devlin, J.**, **M.-W. Chang**, **K. Lee**, and **K. Toutanova** (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. URL `https://arxiv.org/abs/1810.04805`. 2, 33

19. **Elfwing, S.**, **E. Uchibe**, and **K. Doya** (2017). Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. 4, 57, 113

20. **Everingham, M.**, **L. Gool**, **C. K. Williams**, **J. Winn**, and **A. Zisserman** (2010). The pascal visual object classes (voc) challenge. *Int. J. Comput. Vision*, **88**(2), 303–338. ISSN 0920-5691. URL `https://doi.org/10.1007/s11263-009-0275-4`. 27, 51, 69, 87, 107, 123, 137

21. **Glorot, X.** and **Y. Bengio**, Understanding the difficulty of training deep feedforward neural networks. *In* **Y. W. Teh** and **M. Titterington** (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*. JMLR Workshop and Conference Proceedings, Chia Laguna Resort, Sardinia, Italy, 2010. URL `http://proceedings.mlr.press/v9/glorot10a.html`. 28, 69, 88, 108, 125, 138

22. **Goodfellow, I. J.**, **J. Shlens**, and **C. Szegedy** (2014). Explaining and harnessing adversarial examples. URL `https://arxiv.org/abs/1412.6572`. 146

23. **Goodfellow, I. J.**, **D. Warde-Farley**, **M. Mirza**, **A. Courville**, and **Y. Bengio** (2013). Maxout networks. 7, 12, 34, 36

24. **Hahnloser, R.**, **R. Sarpeshkar**, **M. Mahowald**, **R. Douglas**, and **H. Seung** (2000). Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, **405**, 947–51. 112

25. **Hasanpour, S. H.**, **M. Rouhani**, **M. Fayyaz**, and **M. Sabokrou** (2016). Lets keep it simple, using simple architectures to outperform deeper and more complex architectures. URL `https://arxiv.org/abs/1608.06037`. 120, 135

26. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun** (2015*a*). Deep residual learning for image recognition. 20, 26, 43, 44, 53, 61, 64, 81, 86, 103, 120, 135

27. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun** (2015*b*). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. 2, 5, 12, 15, 26, 32, 38, 39, 56, 60, 67, 74, 75, 86, 92, 93, 100, 106, 112, 113, 117, 123, 130, 133, 137

28. **He, K.**, **X. Zhang**, **S. Ren**, and **J. Sun** (2016). Identity mappings in deep residual networks. 20, 43, 61, 64, 72, 81, 90, 99, 103, 120, 135

29. **Hendrycks, D.** and **K. Gimpel** (2020). Gaussian error linear units (gelus). 2, 4, 9, 12, 15, 32, 34, 37, 39, 56, 74, 92, 100, 113, 130, 133

30. **Howard, A. G.**, **M. Zhu**, **B. Chen**, **D. Kalenichenko**, **W. Wang**, **T. Weyand**, **M. Andreetto**, and **H. Adam** (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. URL https://arxiv.org/abs/1704.04861. 20, 43, 81

31. **Hu, J.**, **L. Shen**, **S. Albanie**, **G. Sun**, and **E. Wu** (2017). Squeeze-and-excitation networks. URL https://arxiv.org/abs/1709.01507. 20, 43, 81

32. **Huang, G.**, **Z. Liu**, **L. van der Maaten**, and **K. Q. Weinberger** (2016*a*). Densely connected convolutional networks. 20, 43, 44, 61, 81, 99, 103, 120, 135

33. **Huang, G.**, **Y. Sun**, **Z. Liu**, **D. Sedra**, and **K. Weinberger** (2016*b*). Deep networks with stochastic depth. 120

34. **Hui-zhen Zhao, L.-y. L., Fu-xian Liu** (2017). Improving deep convolutional neural networks with mixed maxout units. 12

35. **Iandola, F. N.**, **S. Han**, **M. W. Moskewicz**, **K. Ashraf**, **W. J. Dally**, and **K. Keutzer** (2016). Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. 20, 43, 81, 121, 135

36. **Ioffe, S.** and **C. Szegedy** (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. 17, 20, 40, 43, 61, 79, 81, 100, 118, 134

37. **Kidger, P.** and **T. Lyons** (2020). Universal approximation with deep narrow networks. 16, 39, 60, 78, 97, 117

38. **Kiefer, J.** and **J. Wolfowitz** (1952). Stochastic estimation of the maximum of a regression function. *Annals of Mathematical Statistics*, **23**, 462–466. 18, 20, 26, 27, 40, 43, 44, 51, 64, 67, 69, 79, 81, 86, 87, 108, 124

39. **Kingma, D. P.** and **J. Ba**, Adam: A method for stochastic optimization. *In* **Y. Bengio** and **Y. LeCun** (eds.), *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 2015. URL http://arxiv.org/abs/1412.6980. 28, 52, 67, 69, 71, 88, 103, 106, 108, 109, 121, 123, 125, 136, 138

40. **Krizhevsky, A.** (2009). Learning multiple layers of features from tiny images. Technical report, University of Toronto. 20, 43, 60, 81, 99, 103, 118, 120, 135

41. **Krizhevsky, A.** (2010). Convolutional deep belief networks on cifar-10. 2, 4, 12, 39, 56, 75, 100

42. **Krizhevsky, A.**, **I. Sutskever**, and **G. E. Hinton**, Imagenet classification with deep convolutional neural networks. *In Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12. Curran Associates Inc., Red Hook, NY, USA, 2012. 17, 20, 43, 61, 64, 79, 81, 99, 100, 103

43. **Kumar, S.**, **K. Biswas**, and **A. K. Pandey**, Predicting landfall's location and time of a tropical cyclone using reanalysis data. *In* **I. Farkaš**, **P. Masulli**, **S. Otte**, and **S. Wermter** (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2021*. Springer International Publishing, Cham, 2021*a*. ISBN 978-3-030-86380-7. iii

44. **Kumar, S.**, **K. Biswas**, and **A. K. Pandey** (2021*b*). Prediction of landfall intensity, location, and time of a tropical cyclone. *Proceedings of the AAAI Conference on Artificial Intelligence*, **35**(17), 14831–14839. URL https://ojs.aaai.org/index.php/AAAI/article/view/17741. ii

45. **Kumar, S.**, **K. Biswas**, and **A. K. Pandey**, Track prediction of tropical cyclones using long short-term memory network. *In 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*. 2021*c*. iii

46. **Kumar, S.**, **K. Biswas**, and **A. K. Pandey** (2022*a*). Forecasting formation of a tropical cyclone using reanalysis data. URL https://doi.org/10.48550/arXiv.2212.06149. iii

47. **Kumar, S.**, **K. Biswas**, and **A. K. Pandey** (2022*b*). Will a tropical cyclone make landfall? *Neural Comput. Appl.*, **35**(8), 5807–5818. ISSN 0941-0643. URL https://doi.org/10.1007/s00521-022-07996-7. iii

48. **Le, Y.** and **X. Yang**, Tiny imagenet visual recognition challenge. 2015. 43, 60, 64, 85, 118

49. **LeCun, Y.**, **B. Boser**, **J. S. Denker**, **D. Henderson**, **R. E. Howard**, **W. Hubbard**, and **L. D. Jackel** (1989). Backpropagation applied to handwritten zip code recognition. *Neural Computation*, **1**(4), 541–551. 15, 38, 39, 60, 78, 92, 97, 117, 133

50. **Lecun, Y.**, **L. Bottou**, **Y. Bengio**, and **P. Haffner** (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, **86**(11), 2278–2324. 17, 20, 43, 61, 81, 99, 100, 103, 121

51. **LeCun, Y.**, **C. Cortes**, and **C. Burges** (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, **2**. 17, 40, 60, 61, 79, 100, 118, 134

52. **Lin, M.**, **Q. Chen**, and **S. Yan** (2014). Network in network. 64

53. **Lin, T.-Y.**, **M. Maire**, **S. Belongie**, **L. Bourdev**, **R. Girshick**, **J. Hays**, **P. Perona**, **D. Ramanan**, **C. L. Zitnick**, and **P. Dollár** (2015). Microsoft coco: Common objects in context. 57

54. **Liu, W.**, **D. Anguelov**, **D. Erhan**, **C. Szegedy**, **S. Reed**, **C.-Y. Fu**, and **A. C. Berg** (2016). Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, 21–37. ISSN 1611-3349. URL http://dx.doi.org/10.1007/978-3-319-46448-0_2. 27, 51, 69, 87, 107, 123

55. **Liu, X.** and **X. Di** (2020). Tanhexp: A smooth activation function with high convergence speed for lightweight neural networks. URL https://arxiv.org/abs/2003.09855. 116

56. **Loshchilov, I.** and **F. Hutter** (2017). Sgdr: Stochastic gradient descent with warm restarts. 18, 40, 43, 64, 79

57. **Ma, N.**, **X. Zhang**, **M. Liu**, and **J. Sun** (2021). Activate or not: Learning customized activation. 2, 12, 56

58. **Ma, N.**, **X. Zhang**, **H.-T. Zheng**, and **J. Sun** (2018). Shufflenet v2: Practical guidelines for efficient cnn architecture design. 20, 26, 43, 61, 64, 67, 81, 86, 103, 120, 135

59. **Maas, A. L.**, **A. Y. Hannun**, and **A. Y. Ng**, Rectifier nonlinearities improve neural network acoustic models. *In in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013*a*. 2, 4, 11, 14, 16, 32, 38, 39, 56, 74, 78, 92, 93, 96, 100, 112, 113, 130, 133

60. **Maas, A. L.**, **A. Y. Hannun**, and **A. Y. Ng**, Rectifier nonlinearities improve neural network acoustic models. *In in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. 2013*b*. 4

61. **Misra, D.** (2020). Mish: A self regularized non-monotonic activation function. 2, 5, 12, 32, 39, 56, 74, 76, 92, 113, 116, 133

62. **Molina, A.**, **P. Schramowski**, and **K. Kersting** (2020). Padé activation units: End-to-end learning of flexible activation functions in deep networks. 2, 6, 12, 16, 32, 39, 56, 59, 74, 75, 78, 93, 94, 95, 96, 97, 100, 116

63. **Nag, S.** and **M. Bhattacharyya** (2021). Serf: Towards better training of deep neural networks using log-softplus error activation function. 56, 59

64. **Nair, V.** and **G. E. Hinton**, Rectified linear units improve restricted boltzmann machines. *In* **J. Fürnkranz** and **T. Joachims** (eds.), *Proceedings of the 27th International Conference on Machine Learning (ICML-10), June 21-24, 2010, Haifa, Israel*. Omnipress, 2010. URL https://icml.cc/Conferences/2010/papers/432.pdf. 2, 3, 11, 12, 14, 32, 39, 56, 74, 92, 100, 112, 113, 129, 133

65. **Negrinho, R.** and **G. Gordon** (2017). Deeparchitect: Automatically designing and training deep architectures. 114, 129

66. **Netzer, Y.**, **T. Wang**, **A. Coates**, **A. Bissacco**, **B. Wu**, and **A. Y. Ng** (2011). Reading digits in natural images with unsupervised feature learning. 17, 40, 60, 61, 79, 100, 118, 119, 135

67. **Nickolls, J.**, **I. Buck**, **M. Garland**, and **K. Skadron**, Scalable parallel programming. *In 2008 IEEE Hot Chips 20 Symposium (HCS)*. 2008. 16, 38, 78, 96

68. **Nwankpa, C.**, **W. Ijomah**, **A. Gachagan**, and **S. Marshall** (2018). Activation functions: Comparison of trends in practice and research for deep learning. 1, 112

69. **Paszke, A.**, **S. Gross**, **F. Massa**, **A. Lerer**, **J. Bradbury**, **G. Chanan**, **T. Killeen**, **Z. Lin**, **N. Gimelshein**, **L. Antiga**, **A. Desmaison**, **A. Köpf**, **E. Yang**, **Z. DeVito**, **M. Raison**, **A. Tejani**, **S. Chilamkurthy**, **B. Steiner**, **L. Fang**, **J. Bai**, and **S. Chintala** (2019). Pytorch: An imperative style, high-performance deep learning library. 16, 38, 78, 96, 134

70. **Qiu, S.**, **X. Xu**, and **B. Cai** (2017). Frelu: Flexible rectified linear units for improving convolutional neural networks. 113

71. **Radford, A.**, **J. Wu**, **R. Child**, **D. Luan**, **D. Amodei**, and **I. Sutskever** (2019). Language Models are Unsupervised Multitask Learners. URL https://openai.com/blog/better-language-models/. 2, 33, 57

72. **Radosavovic, I.**, **R. P. Kosaraju**, **R. Girshick**, **K. He**, and **P. Dollár** (2020). Designing network design spaces. 120

73. **Ramachandran, P.**, **B. Zoph**, and **Q. V. Le** (2017). Searching for activation functions. 2, 5, 9, 11, 12, 32, 39, 56, 57, 74, 75, 92, 93, 100, 113, 114, 129, 130, 133

74. **Robbins, H.** and **S. Monro** (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, **22**, 400–407. 18, 20, 26, 27, 40, 43, 44, 51, 64, 67, 69, 79, 81, 86, 87, 107, 124

75. **Ronneberger, O.**, **P. Fischer**, and **T. Brox** (2015). U-net: Convolutional networks for biomedical image segmentation. 28, 52, 69, 87, 108, 125, 138

76. **Sandler, M.**, **A. Howard**, **M. Zhu**, **A. Zhmoginov**, and **L.-C. Chen** (2019). Mobilenetv2: Inverted residuals and linear bottlenecks. 20, 43, 61, 81, 103, 104, 120, 135

77. **Simonyan, K.** and **A. Zisserman** (2015). Very deep convolutional networks for large-scale image recognition. 17, 20, 27, 40, 43, 51, 61, 69, 79, 81, 87, 100, 103, 120, 135

78. **Singh, Y.** and **P. Chandra** (2003). A class +1 sigmoidal activation functions for ffanns. *Journal of Economic Dynamics and Control*, **28**, 183–187. 114

79. **Srivastava, N.**, **G. Hinton**, **A. Krizhevsky**, **I. Sutskever**, and **R. Salakhutdinov** (2014). Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, **15**(1), 1929–1958. ISSN 1532-4435. 26, 28, 52, 61, 67, 71, 86, 88, 100, 106, 109, 118, 123, 125, 134, 137

80. **Szegedy, C.**, **W. Liu**, **Y. Jia**, **P. Sermanet**, **S. Reed**, **D. Anguelov**, **D. Erhan**, **V. Vanhoucke**, and **A. Rabinovich** (2014*a*). Going deeper with convolutions. 43, 64, 103

81. **Szegedy, C.**, **W. Liu**, **Y. Jia**, **P. Sermanet**, **S. Reed**, **D. Anguelov**, **D. Erhan**, **V. Vanhoucke**, and **A. Rabinovich** (2014*b*). Going deeper with convolutions. 120

82. **Szegedy, C.**, **V. Vanhoucke**, **S. Ioffe**, **J. Shlens**, and **Z. Wojna** (2015*a*). Rethinking the inception architecture for computer vision. 20, 43, 61, 81, 120, 135

83. **Szegedy, C.**, **V. Vanhoucke**, **S. Ioffe**, **J. Shlens**, and **Z. Wojna** (2015*b*). Rethinking the inception architecture for computer vision. 103

84. **Tan, M.** and **Q. V. Le** (2020). Efficientnet: Rethinking model scaling for convolutional neural networks. 20, 43, 61, 81, 99, 103, 120

85. **Targ, S.**, **D. Almeida**, and **K. Lyman** (2016). Resnet in resnet: Generalizing residual architectures. 120

86. **Teh, Y. W.** and **G. E. Hinton**, Rate-coded restricted boltzmann machines for face recognition. *In Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00. MIT Press, Cambridge, MA, USA, 2000. 97

87. **Trottier, L.**, **P. Giguère**, and **B. Chaib-draa** (2017). Parametric exponential linear unit for deep convolutional neural networks. *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 207–214. 113

88. **Vaswani, A.**, **N. Shazeer**, **N. Parmar**, **J. Uszkoreit**, **L. Jones**, **A. N. Gomez**, **L. Kaiser**, and **I. Polosukhin** (2017). Attention is all you need. 28, 52, 53, 71, 88, 109, 125, 139

89. **Weisstein, E. W.** (). Orthogonal polynomials. URL https://mathworld.wolfram.com/OrthogonalPolynomials.html. 94, 95

90. **Xiao, H.**, **K. Rasul**, and **R. Vollgraf** (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*. 17, 40, 60, 61, 79, 100, 118, 119, 134

91. **Xie, S.**, **R. Girshick**, **P. Dollár**, **Z. Tu**, and **K. He** (2017). Aggregated residual transformations for deep neural networks. 20, 43, 64, 81, 103, 120

92. **Xu, B.**, **N. Wang**, **T. Chen**, and **M. Li** (2015*a*). Empirical evaluation of rectified activations in convolutional network. URL https://arxiv.org/abs/1505.00853. 32, 74, 113, 130

93. **Xu, B.**, **N. Wang**, **T. Chen**, and **M. Li** (2015*b*). Empirical evaluation of rectified activations in convolutional network. 113

94. **Xu, Y.** and **H. Zhang** (2021). Convergence of deep relu networks. URL https://arxiv.org/abs/2107.12530. 145

95. **Yu, F.**, **D. Wang**, **E. Shelhamer**, and **T. Darrell** (2019). Deep layer aggregation. 64, 103, 120

96. **Zagoruyko, S.** and **N. Komodakis** (2016). Wide residual networks. 20, 26, 43, 44, 61, 64, 81, 86, 103, 106, 120, 123, 135, 137

97. **Zhang, H.**, **M. Cisse**, **Y. N. Dauphin**, and **D. Lopez-Paz** (2017*a*). mixup: Beyond empirical risk minimization. URL https://arxiv.org/abs/1710.09412. 20, 64, 81

98. **Zhang, X.**, **X. Zhou**, **M. Lin**, and **J. Sun** (2017*b*). Shufflenet: An extremely efficient convolutional neural network for mobile devices. URL https://arxiv.org/abs/1707.01083. 43

99. **Zheng, H.**, **Z. Yang**, **W. Liu**, **J. Liang**, and **Y. Li**, Improving deep neural networks using softplus units. *In 2015 International Joint Conference on Neural Networks (IJCNN)*. 2015. 4, 12, 32, 39, 74, 75, 92, 133

100. **Zhou, Y.**, **D. Li**, **S. Huo**, and **S.-Y. Kung** (2020). Soft-root-sign activation function. 112, 113, 129, 130

101. **Zoph, B.**, **V. Vasudevan**, **J. Shlens**, and **Q. V. Le** (2018). Learning transferable architectures for scalable image recognition. 120