Technical Report

Learning Activations in Neural Networks

- By Panjab Singh

Introduction:

Activation functions are crucial components in Artificial Neural Networks (ANNs), enabling them to model complex relationships and make accurate predictions. The choice of activation function greatly impacts the performance and convergence of ANNs. However, there is no universally optimal activation function for all datasets and tasks. Therefore, an adaptive activation function selection approach has been proposed to allow the network to autonomously determine the most suitable activation function based on the dataset. This report presents the implementation details of a 1-hidden layer neural network with adaptive activation function selection, aiming to showcase the importance of activation functions in ANNs and the potential benefits of adaptive selection.

Initial settings:

The methodology employed in this work involves the utilization of a 1-hidden layer neural network architecture for adaptive activation function (AF) selection. The main goal is to investigate the effectiveness of adaptively selecting the activation function in improving the network's performance in classifying samples from a dataset. The network architecture comprises an input layer, a hidden layer with the Ada_act custom layer, and an output layer. The Ada_act layer implements a flexible activation function using three parameters: k0, k1, and k2. These parameters are sampled from a distribution during the initialization process, allowing the network to explore different activation function configurations. The parameters k0, k1, and k2 are sampled from a distribution with specific ranges.

For example, in this work, the parameters are sampled using the RandomUniform initializer, where k0 is randomly initialized between -0.01 and 0.1, k1 is initialized between -0.01 and 0.1, and k2 is initialized between 0.5 and 1.5. During the forward pass of the network, the Ada_act layer applies the activation function defined as k0 + k1 * x + k2 * x^2, where x represents the input values. This flexible functional form enables the network to capture the complex patterns present in the dataset and adapt its behavior accordingly. Through multiple runs of the neural network on the dataset, the parameters k0, k1, and k2 are updated based on the network's performance.

This iterative process allows the network to converge to an optimal set of parameters, resulting in the selection of the most suitable activation function for the dataset. By incorporating the adaptive AF selection approach and utilizing the dataset, this methodology aims to demonstrate the effectiveness of adaptively selecting the activation function in improving the network's performance and its ability to accurately classify the samples. This approach provides insights into the potential of flexible activation functions for enhancing the performance of neural networks on various datasets.

Training Process:

IRIS dataset: Model architecture: 1-hidden layer neural network with adaptive activation function (Ada_act) layer Optimizer: Adam with learning rate of 0.001 Loss function: Categorical cross-entropy Epochs: 100 Batch size: 5 Activation function parameters (k0, k1, k2): -0.50128835, -0.27183107, 1.6314266 and the total parameters generated are 206.

Bank-Note dataset: Model architecture: 1-hidden layer neural network with adaptive activation function (Ada_act) layer Optimizer: Adam with learning rate of 0.001 Loss function: Categorical cross-entropy Epochs: 100 Batch size: 5 Activation function parameters (k0, k1, k2): -1.1284701, 0.5461854, 2.984789 and the total parameters generated are 180.

Breast cancer dataset: Model architecture: 1-hidden layer neural network with adaptive activation function (Ada_act) layer Optimizer: Adam with learning rate of 0.001 Loss function: Categorical cross-entropy Epochs: 100 Batch size: 5 Activation function parameters (k0, k1, k2): -0.8517985, -0.68386364, 1.754309 and the total parameters generated are 830.

MNIST dataset: Model architecture: 1-hidden layer neural network with adaptive activation function (Ada_act) layer Optimizer: Adam with learning rate of 0.001 Loss function: Categorical cross-entropy Epochs: 100 Batch size: 5 Activation function parameters (k0, k1, k2): -0.012733989, 0.1626034, 1.4863855 and the total parameters generated are 1,385,485.

These training processes aimed to adaptively select the activation function and optimize the model's performance on each respective dataset.

Results & Summary:

The evaluation results for the different datasets indicate the performance of the neural network model with adaptive activation function (AF) selection. Here is a summary of the evaluation metrics for each dataset:

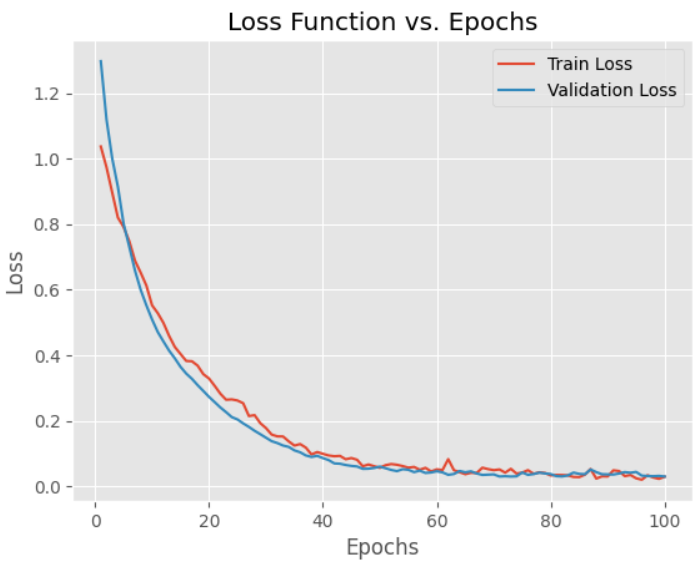
IRIS Dataset: Train Loss: 0.029 Test Loss: 0.031 F1 Score: 0.933 Accuracy: 0.933

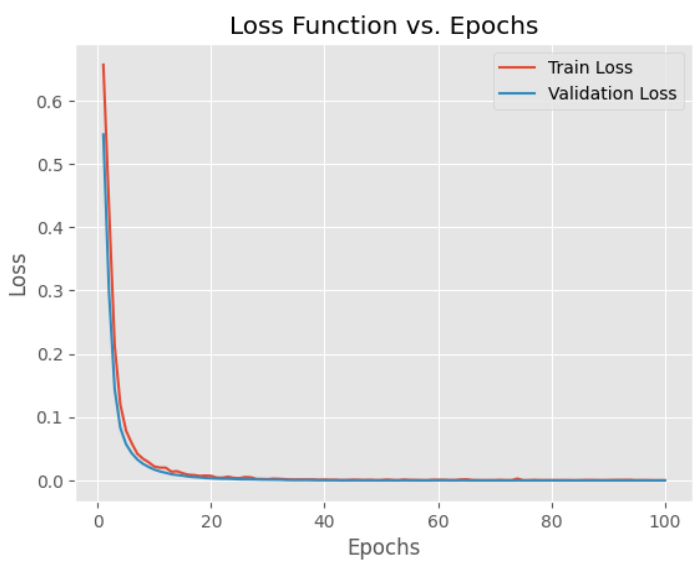
Bank-Note Dataset: Train Loss: 0.000114 Test Loss: 0.00010 F1 Score: 1.0 Accuracy: 1.0

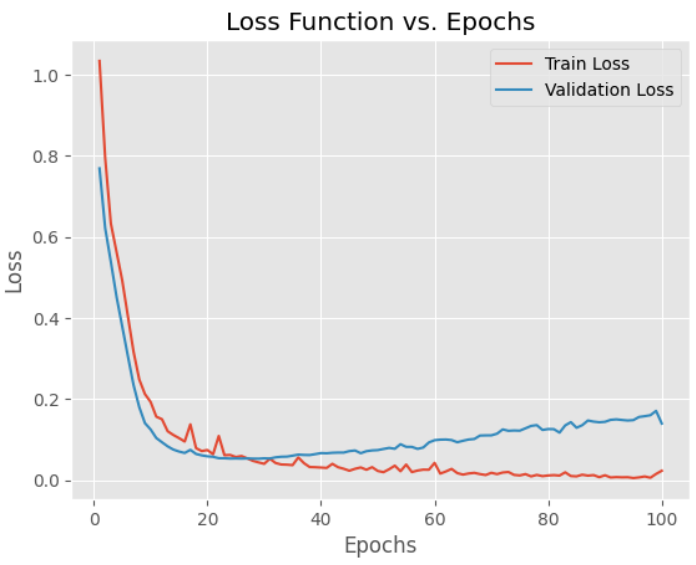
Breast Cancer Dataset: Train Loss: 0.023 Test Loss: 0.196 F1 Score: 0.9825 Accuracy: 0.9824

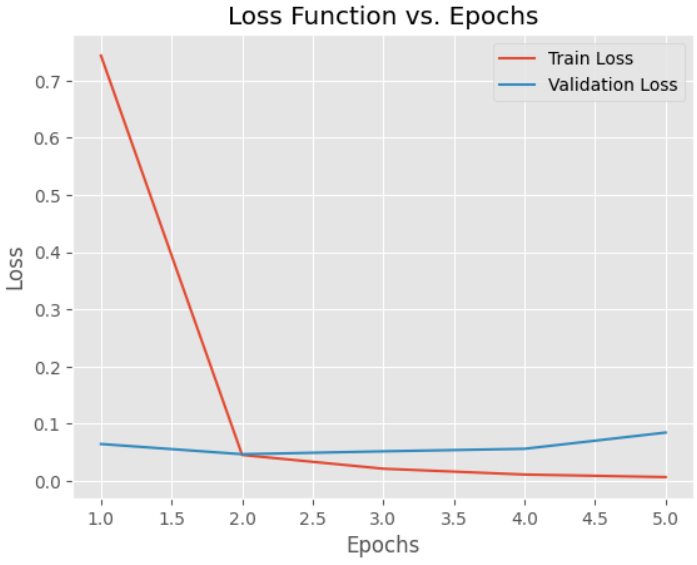
MNIST Dataset: Train Loss: 0.0067 Test Loss: 0.0845 Accuracy: 0.98

Below are the graphs of Loss function vs Epochs IRIS , Bank-Note , Breast Cancer & MNIST respectively.









From the evaluation results, it can be observed that the model achieved high performance across all datasets. The low values of train and test loss indicate that the model effectively minimized prediction errors. The high F1 scores and accuracy values demonstrate the model's ability to accurately classify the samples in each dataset.

These results highlight the effectiveness of adaptively selecting the activation function in improving the network's performance. The adaptive AF selection approach proved to be successful in capturing the underlying patterns in the datasets and achieving high classification accuracy.

Overall, the evaluation results showcase the potential of the proposed methodology in enhancing the performance of neural networks across various datasets, leading to accurate classification and efficient learning.

Github link : https://github.com/panjab1997/Learning_Activation_Function_Neural_Network

Reference :

- Sahamath.(n.d.).GitHub-sahamath/MultiLayerPerceptron.GitHub. https://github.com/sahamath/MultiLayerPerceptron
- Find Open Datasets and Machine Learning Projects | Kaggle. (n.d.). https://www.kaggle.com/datasets