



Hyperparameter Tuning for Deep Learning Neural Networks



Team Members:

- Martin Luis Macho
 - Jacob Roman
 - Jose Lopez
-
- Product Owner: Yanzhao Wu
 - Instructor: Dr. Masoud Sadjadi





Introduction

Machine Learning is a current popular subject in the field of computing, thanks to advancements in computational resources and power, as well as the ease of access with libraries such as TensorFlow and Keras. With greater ease of access, more people are able to train incredibly complex models, such as neural networks, very quickly with little experience. However, due to this lack of experience, many of the hyperparameters used to tune machine learning models can be difficult to understand.

The current system is LRBench, a program that automatically tunes the learning rate function of a trained model to optimize training accuracy. This system was developed to research how to choose the best learning rate function for training a model. (Wu, 2023)

The new system is intended to do a similar process as LRBench for batch size, another Hyperparameter that can greatly impact accuracy and speed of training, but can be difficult to optimize manually without a fine and detailed understanding of how the value impacts training.





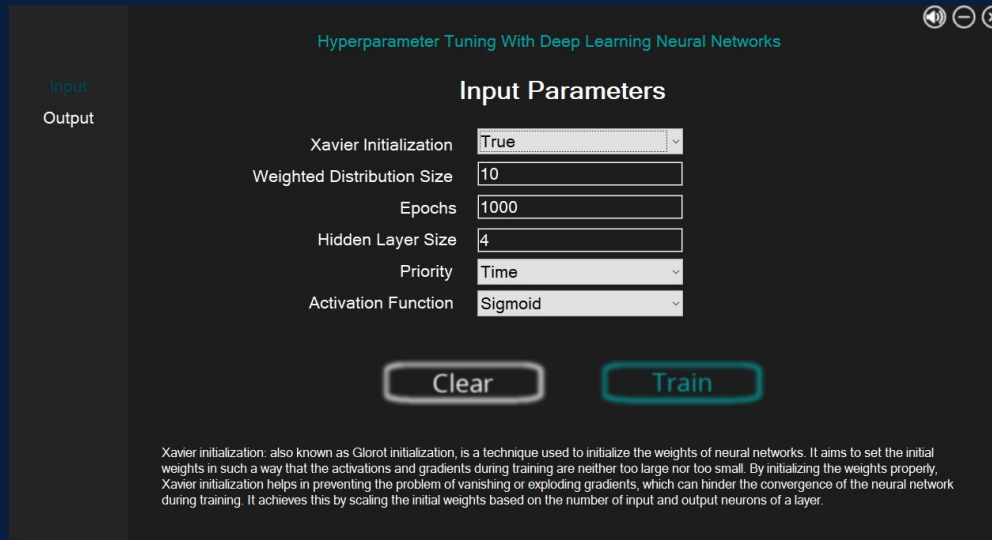
User Stories

- User Story 1: Background Study and Research
- User Story 2: Plan based on Background Research
- User Story 3: Study existing project (LRBench)
- User Story 4: Write project proposal
- User Story 5: Develop backend for product
- User Story 6: Study and experiment with LRBench project to guide backend development
- User Story 7: Develop and test rudimentary algorithm for batch size tuning
- User Story 8: Continue working on backend for product
- User Story 9: Design and develop frontend UI/functionality for product
- User Story 10: Finalize product prototype
- User Story 11: Test product prototype
- User Story 12: Complete documentation on project



User Interface

Input



Hyperparameter Tuning With Deep Learning Neural Networks

Input
Output

Input Parameters

Xavier Initialization:

Weighted Distribution Size:

Epochs:

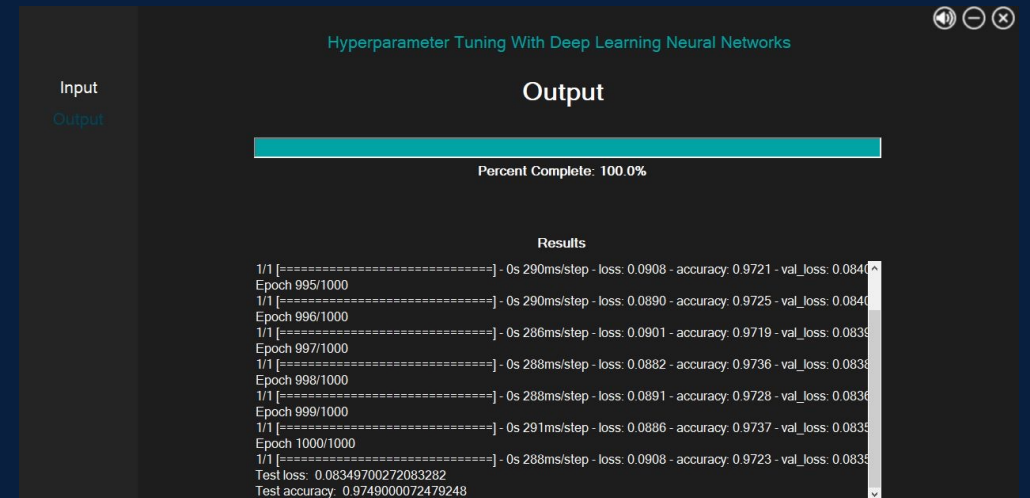
Hidden Layer Size:

Priority:

Activation Function:

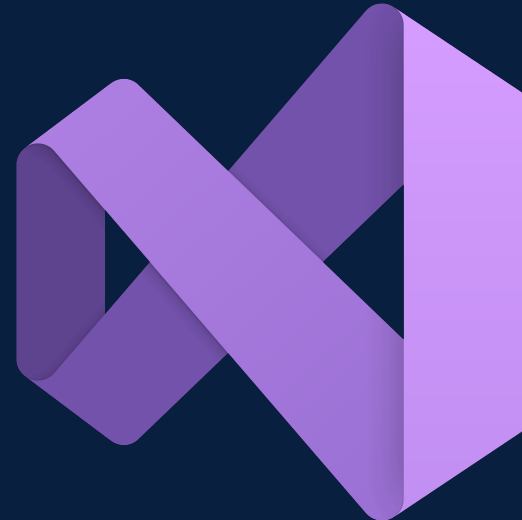
Xavier initialization: also known as Glorot initialization, is a technique used to initialize the weights of neural networks. It aims to set the initial weights in such a way that the activations and gradients during training are neither too large nor too small. By initializing the weights properly, Xavier initialization helps in preventing the problem of vanishing or exploding gradients, which can hinder the convergence of the neural network during training. It achieves this by scaling the initial weights based on the number of input and output neurons of a layer.

Output



Technologies Used

- Python
- C#
- Windows Forms
- Visual Studio 2022
- Visual Studio Code



Shortcomings

- The system as currently designed is not very portable; future iterations would move away from Windows Forms for ease of installation on different operating systems.
- The system currently only operated on one neural network model and one dataset; future iterations should focus on expanding the system's capabilities to multiple datasets.
- The algorithm used to tune the batch size is still rudimentary; it only tunes the batch size once at the start of training, and uses a simple algorithm based on basic experimentation. Future iterations would ideally update batch size dynamically based on training performance on multiple datasets and models.



Thank you!