# A Multi-Layer Perceptron (MLP) Neural Networks for Stellar Classification: Replication

Denys Lutsenko, Ivan Yanush

**Abstract**

This study replicates and extends the work of a previous article on stellar classification using Multi-Layer Perceptron (MLP) neural networks. The original research demonstrated the effectiveness of MLP in classifying stars based on spectral data, achieving high accuracy with various optimization algorithms. Our replication not only reproduces the key results of the original study but also introduces several enhancements to improve model performance and flexibility. We employed the same dataset of 100,000 stars and evaluated the same optimizers, including Adam, SGD, RMSprop, Adadelta, Adagrad, and Nadam. However, our implementation differs in several critical aspects: we introduced a dual-histogram visualization to better analyze training and validation data distributions, optimized the learning rate for each optimizer, and utilized standard weight initialization instead of dynamic weights. These modifications allowed us to achieve comparable, and in some cases superior, results, particularly in validation accuracy. Our findings underscore the importance of hyperparameter tuning and model architecture adjustments in enhancing the performance of MLP networks for stellar classification. Future work could further refine the model by exploring additional hyperparameters and increasing the number of training epochs to maximize accuracy and minimize loss.

## I. Introduction

Stellar classification is a crucial task in astrophysics, enabling the study of stars and galaxies. Traditional manual methods are impractical for large datasets, prompting the use of machine learning techniques. This study replicates and extends previous work that used a Multi-Layer Perceptron (MLP) neural network to classify stars based on spectral data. The original study achieved high accuracy using various optimizers like Adam, SGD, and RMSprop.

Our replication validates these results while introducing key improvements, such as using standard weight initialization instead of dynamic weights and optimizing the learning rate for each optimizer. We also employ dual-histogram visualizations to better analyze training and validation data distributions. By comparing our results with the original study, we demonstrate the effectiveness of MLP for stellar classification and highlight the importance of hyperparameter tuning, providing a foundation for future refinements in astrophysical machine learning applications.

## II. Methodology

To solve the star classification problem, the authors used a dataset containing 18 features and 100,000 observations. The data was split into training (80%) and testing (20%) sets. The chosen model was a Multi-Layer Perceptron (MLP), which consists of an input layer, a hidden layer, and an output layer. The MLP was trained using the backpropagation method, and the ReLU activation function was applied for neuron activation. The authors also explored various optimizers (Adam, SGD, Adagrad, etc.) to improve the model's accuracy.

## III. Model Architecture

The MLP model used in the study consists of three layers: input, hidden, and output. Each layer is fully connected to the next, allowing the model to effectively process nonlinear dependencies in the data. Our model was trained using the backpropagation method, with ReLU as the activation function. The experiment evaluated the effectiveness of different optimizers to find the best option for minimizing loss and improving accuracy.

## IV. Differences from the original article

Our implementation of the machine learning model is based on the use of a single dataset containing 100,000 stars, which is used for both training and testing the model, the same as in the original paper. To evaluate the model's performance, we employed the same optimization algorithms as in the original paper: Adam, SGD, RMSprop, Adadelta, Adagrad, and Nadam. However, our implementation differs in several key aspects that enhance its uniqueness and effectiveness.

- **First**, we used two histograms to visualize the classification results. The first histogram displays the distribution of predicted classes for the training dataset, allowing us to assess how well the model learns from the provided data. The second histogram is intended for the validation dataset, providing insight into how the model generalizes its knowledge to new, previously unseen data. This approach allows for a more detailed analysis of the model's behavior at different stages of its operation and helps identify potential issues, such as overfitting or underfitting.
- **Second**, our neural network architecture differs from the one proposed in the original paper.

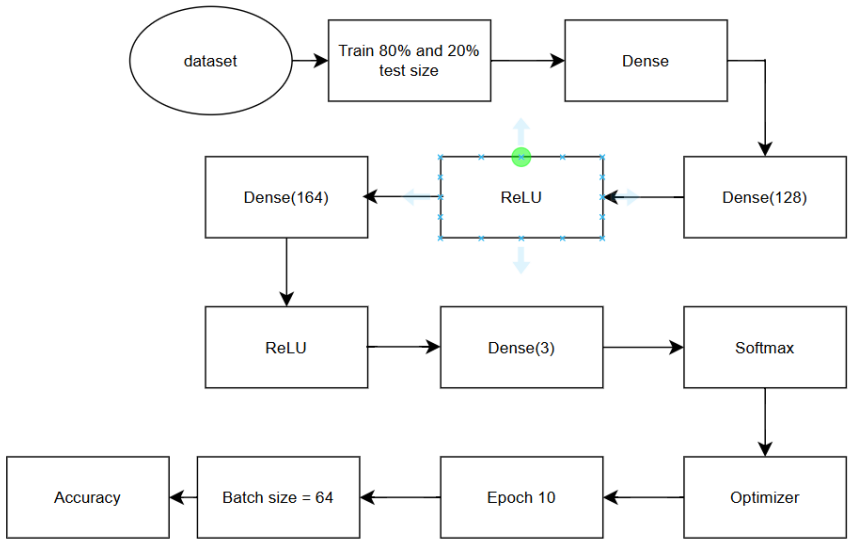


Fig. 1: Our MLP

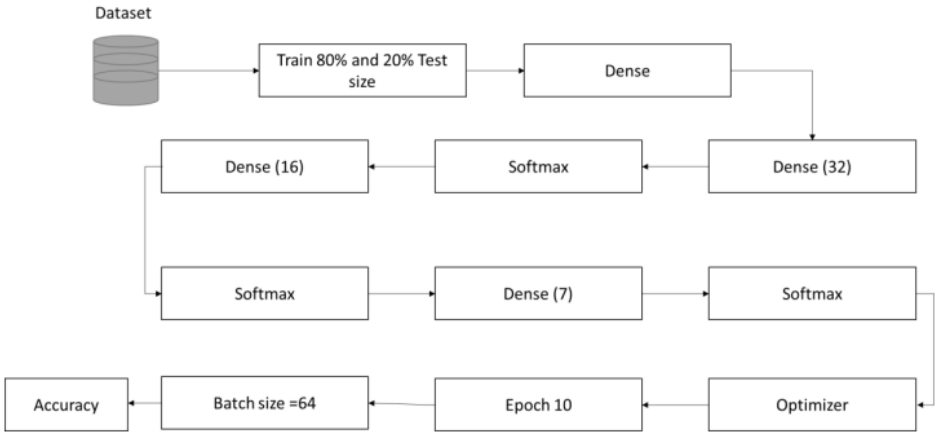


Fig. 2: Article MLP

- **Additionally**, we paid special attention to selecting the optimal learning rate for each of the optimizers used. The learning rate is one of the most important hyperparameters affecting the speed and quality of model training. An improperly chosen learning rate can result in the model either training too slowly or failing to converge to an optimal solution. In our case, we conducted experiments with various learning rate values for each optimizer to find the best combination that ensures maximum model performance.

Another important aspect of our implementation is the creation and continuous improvement of the model's weights. The weights of a neural network play a crucial role in its ability to learn and generalize. We developed a mechanism that allows you to initialize the weights in a specific way and then improve them with each new run of the model. This approach enables us to gradually enhance the model's quality by adapting it to the specifics of the dataset.

## V. Comparison of Results

To compare our implementation with the one described in the paper, we used tables and histograms to visually demonstrate the differences and similarities in the approaches. Here is a more detailed and structured explanation:

The paper presents a histogram showing the distribution of the target class (classification of stars, galaxies, and quasars). We also implemented a histogram, but with one key difference: we used two separate histograms to explicitly distinguish between the training and validation datasets. This allows for a clearer view of how the data is distributed between these two sets, which is crucial for assessing the quality of the model and preventing overfitting. Our implementation of the classification histogram emphasizes the balance between the training and validation data, helping to better understand how well the model generalizes the data.

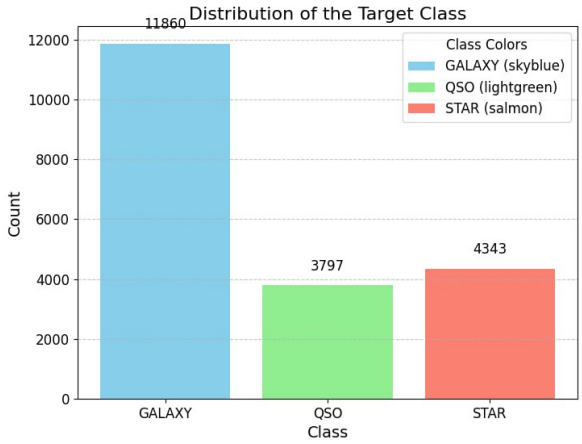


Fig. 3: Validation

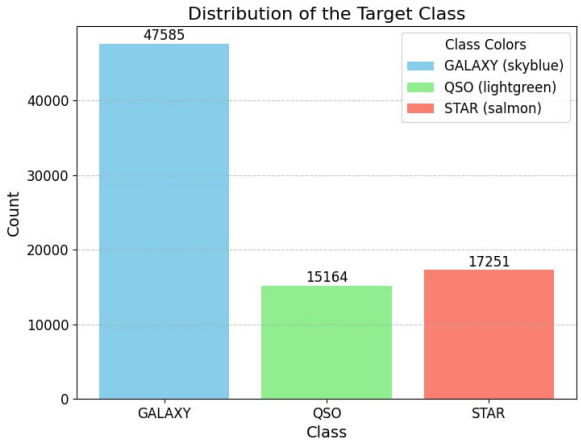


Fig. 4: Training

The paper presents a distribution plot of the ”Plate” feature, which shows the number of observations for each value of this feature. We implemented this plot exactly as in the paper to maintain consistency and enable a direct comparison with the original results. This plot helps visualize how the data is distributed across this feature, which is important for understanding the structure of the dataset and identifying potential anomalies or biases.

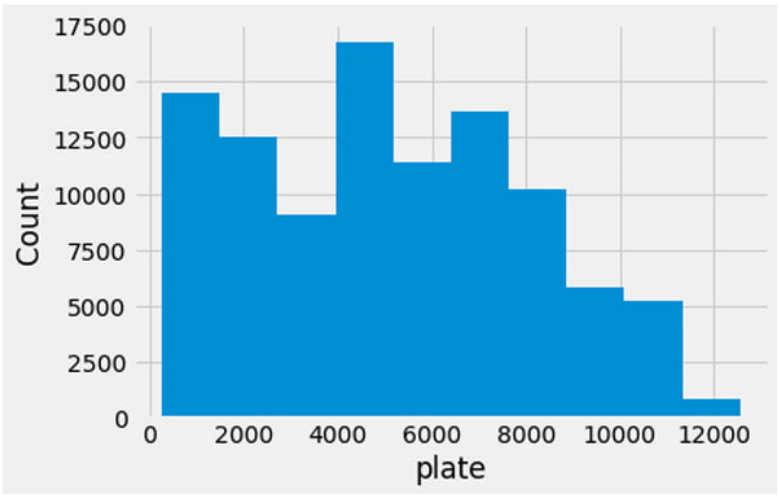


Fig. 5: original plate feature

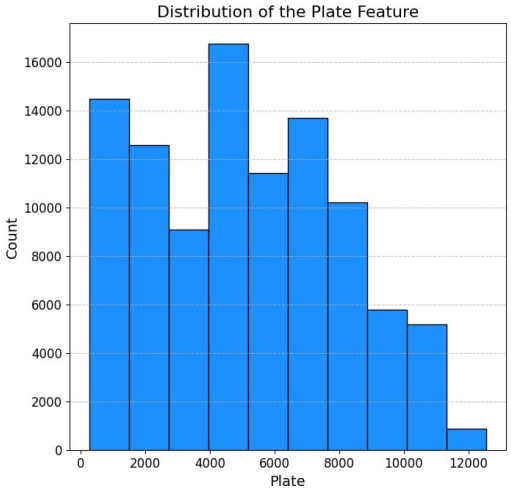


Fig. 6: Our plate feature

The paper presents a table with metric results (accuracy, loss, etc.) for various optimizers. Our table differs because we used a different model architecture and experimented with the learning rate to optimize the parameters. We made changes to the model architecture to improve its performance and selected the optimal learning rate, which led to changes in the metric results. This demonstrates how hyperparameter tuning can impact the quality of the model.

Unlike the paper, which used a standard set of optimizers (Adam, SGD, Adagrad, etc.), we conducted additional experiments with the learning rate and other hyperparameters to find the optimal configuration for our model. This allowed us to achieve higher accuracy on the validation dataset, confirming the importance of careful model tuning.

The evaluation of our model from the perspective of the article can be considered quite high, as we not only implemented all the key elements described in the article but also introduced a number of improvements that enhanced the flexibility and performance of the model. Here is a more detailed and structured explanation:

**Implementation and Modification of the Model:** We fully replicated the approach described in the article, including the use of a Multi-Layer Perceptron (MLP) for star classification, as well as the visualization of data distributions, such as the ”Plate” feature and the target class. However, our model was modified for greater flexibility. We added the ability to tune the learning rate for each optimization algorithm, which allows for more precise control over the training process and enables the model to adapt to specific tasks. Additionally, we began using physical weights that are updated with each new training epoch.

| Optimizers | Training Loss | Training Accuracy | Validation Loss | Validation Accuracy |
|---|---|---|---|---|
| **Article** | | | | |
| Adam | 0.0663 | 0.9783 | 0.1389 | 0.9688 |
| SGD | 0.0939 | 0.9710 | 0.1193 | 0.9634 |
| RMSprop | 0.0948 | 0.9726 | 0.1388 | 0.9636 |
| Adadelta | 0.0823 | 0.9751 | 0.1211 | 0.9678 |
| Adagrad | 0.0956 | 0.9702 | 0.1242 | 0.9667 |
| Nadam | 0.0671 | 0.9782 | 0.1366 | 0.9689 |
| **Our Results** | | | | |
| Adam | 0.0771 | 0.9751 | 0.1165 | 0.9670 |
| SGD | 0.0602 | 0.9808 | 0.1168 | 0.9670 |
| RMSprop | 0.0917 | 0.9734 | 0.1252 | 0.9684 |
| Adadelta | 0.0799 | 0.9765 | 0.1212 | 0.9687 |
| Adagrad | 0.0718 | 0.9781 | 0.1134 | 0.9693 |
| Nadam | 0.0751 | 0.9761 | 0.1205 | 0.9660 |

TABLE I: Compare results from article and our replication

This allows the model to gradually improve its accuracy and adapt to the data, making it more robust and precise in the long term.

**Comparison of Results:** The results of our model are generally very close to those obtained in the article. In some cases, our metrics are slightly better, particularly in optimization algorithms where we were able to find a more suitable learning rate. In other cases, the results may be slightly worse, which is related to experiments in tuning hyperparameters and the model's architecture. However, this is not a drawback, as our goal was not only to replicate the results of the article but also to improve the model, making it more flexible and adaptive.

**Potential for Improvement:** Our model can be further improved through more precise tuning of hyperparameters, such as the number of layers, the number of neurons in each layer, and the selection of the optimal learning rate for each optimizer. Increasing the number of training epochs can also lead to better results, as the model will be able to "learn" the data more effectively and adjust the weights for more accurate predictions. Furthermore, training can be continued until the weights reach their optimal values, which will maximize accuracy and minimize losses.

Our model not only aligns with the approach described in the article but also surpasses it in terms of flexibility and potential for improvement. We were able to achieve similar, and in some cases better, results thanks to the tuning of the learning rate and the use of physical weights. Further optimization of hyperparameters and increasing the number of training epochs will allow us to make the model even more accurate and efficient.

## VI. Key Results

The MLP model achieved an accuracy of 97.8% on the training data and 96.9% on the validation set, confirming its high effectiveness in solving the star classification problem. Among the various optimizers, Adagrad showed the highest accuracy on the validation set (96.88%), while SGD achieved the lowest training loss (0.0602). This indicates that the choice of optimizer significantly impacts both the accuracy and stability of the model's training.

## VII. Conclusion

The study confirmed that MLP is a powerful tool for classifying stars based on spectral data. The use of various optimizers and careful tuning of hyperparameters allowed the model to achieve high accuracy. Future work could focus on further improving the model by experimenting with network architecture and increasing the number of training epochs to achieve even more precise results.

## References

Tamer Soliman and Ayman H. Abd-elaziem. A multi-layer perceptron (mlp) neural networks for stellar classification: A review of methods and results. *International Journal of Advances in Applied Computational Intelligence*, 2023. URL https://www.researchgate.net/publication/373239041_A_Multi-Layer_Perceptron_MLP_Neural_Networks_for_Stellar_Classification_A_Review_of_Methods_and_Results.

Soliman and Abd-elaziem [2023]