Improving the Accuracy of a Neural Network

Javier A. Diaz Velazquez

Colorado State University Global

CSC580: Applying Machine Learning and Neural Networks

Dr. Issa

August 20, 2023

Tox21 Model Improvement

In order to be able to perform the test and the analysis required per the assignment instruction, it was necessary to improve the model and add some extra features that made the testing task easy to perform. One of the requirements for this assignment was to add a random forest classifier using the sklearn python library to run the Tox21 data set through the random forest and record its performance accuracy, which later on shall be used for comparison. The hyperparameters of the random forest classifier were the following; 50 hidden layers, 0.001 learning rate, ten epochs, 100 batches size, and 0.5 dropout probe. Such hyperparameter configuration yielded a weighted train accuracy of 99%, a weighted valid accuracy of 69%, and a weighted accuracy of 67%, which are pretty good compared with the accuracy of the previous TensorFlow model, which yielded a valid accuracy of 50% and a test accuracy of 53% when utilizing similar hyperparameters setting.

Once that code was functional, the second parameter per the assignment instructions was to select a list of hyperparameter values needed to run the algorithm to see which of the tuning settings allows for better performance and yields the best accuracy results. However, it was necessary to integrate user interaction into the algorithm to make the process's testing and analysis phase more streamlined and controllable, meaning that instead of hardcoding each value and executing the code over and over with different hyperparameter values, an implementation of an input method was integrated within the nested for loop which enables a prompt that allows the entry of different hyperparameter values during execution which iterates for the following hyperparameter values after completing each execution test. Figures 1 – 1 and 1 – 2 show the input method's implementation and the prompt during execution.

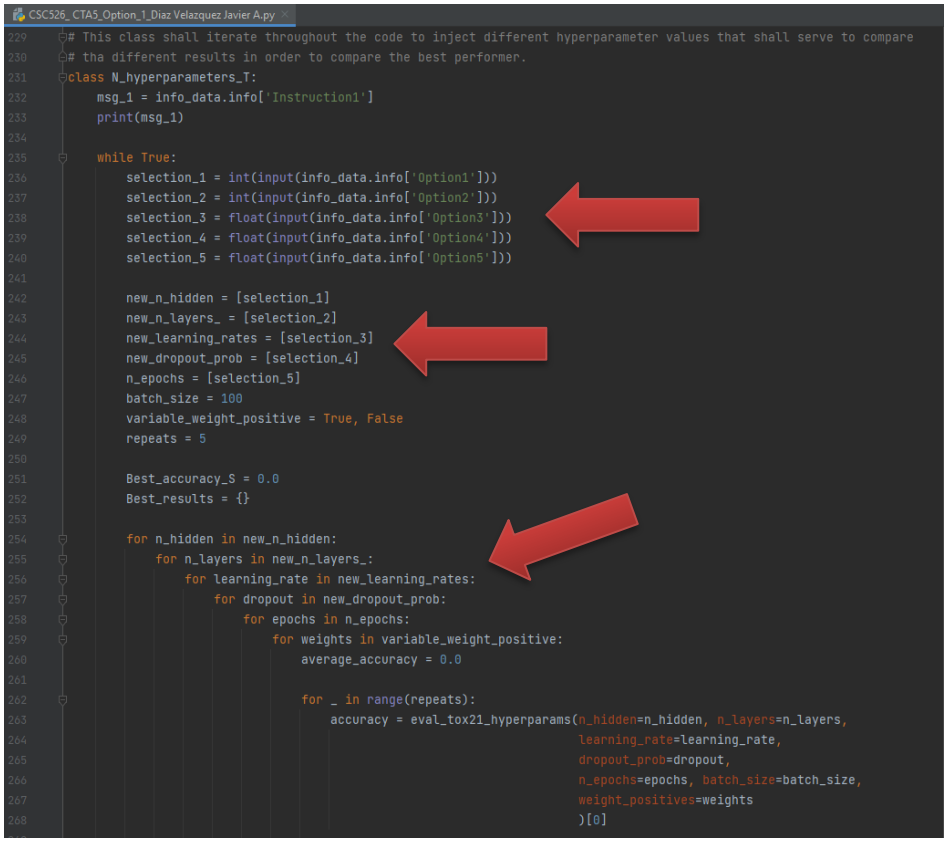

Figure 1 – 1 The input method's implementation and the nested for loop.

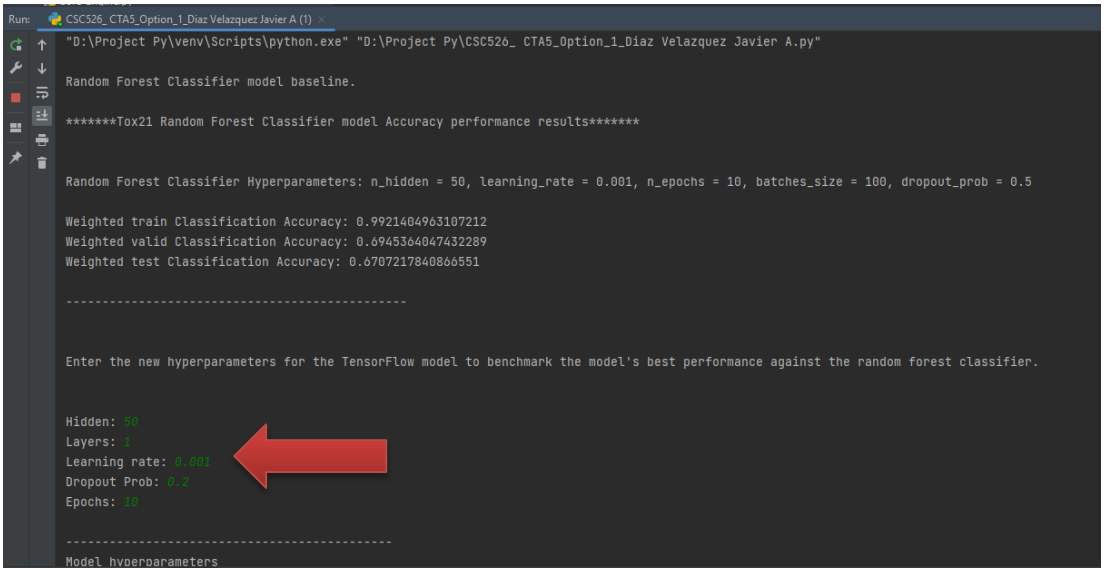

Figure 1 – 2 Input method prompt during execution allows for a better control flow for analysis.

Tox21 Model's Performance Accuracy

The Tox21 TensorFlow model was tested and analyzed with sets of different hyperparameter tuning settings to achieve and yield the best performance accuracy. The first execution run was performed with the following hyperparameters; 10 hidden, one layer, learning rate at 0.0001, the dropout probe at 0.1, and ten epochs. The resulting accuracy performance, given the values of the hyperparameters, yielded disappointing results compared with the results of the random forest classifier. The model yielded 50% of train-weighted classification accuracy, 45% of test-weighted classification accuracy, and 45% of valid weighted classification accuracy, with an average of 13% overall accuracy. The second test was performed with the following hyperparameters; 50 hidden, one layer, learning rate at 0.001, dropout probe 0.2, and ten epochs. This time the result was far better than the first execution; the training accuracy was 53%, testing accuracy 55%, and valid classification accuracy at 50%, a Wapping 96% of average accuracy, which was far superior to the previous results.

The hyperparameters of the third execution test run were 100 hidden, two layers, learning rate at 0.01, dropout at 0.5, and 100 epochs. The results show an improvement of 92% in train-weighted classification accuracy, 64% in testing classification accuracy, and a valid accuracy of 58%, yielding an average of 96% accuracy. The fourth hyperparameter values for this execution run were 300 hidden, four layers, learning rate 1.0, dropout 0.9, and 100 epochs. Now the expectation for this particular hyperparameter setting was 50/50 chances of either a better execution or the worst execution than the previous results. The results were as follows; the training accuracy dropped by 3%, yielding a 50% accuracy; the testing accuracy dropped by 1%, yielding a 54% accuracy; the valid accuracy stays at 50% with no loss resulting in an overall accuracy of 96% which is not bad. Figures 1 – 3 and 1 – 4 demonstrated the different hyperparameter values, the execution, the results, and the TensorBoard graphs.

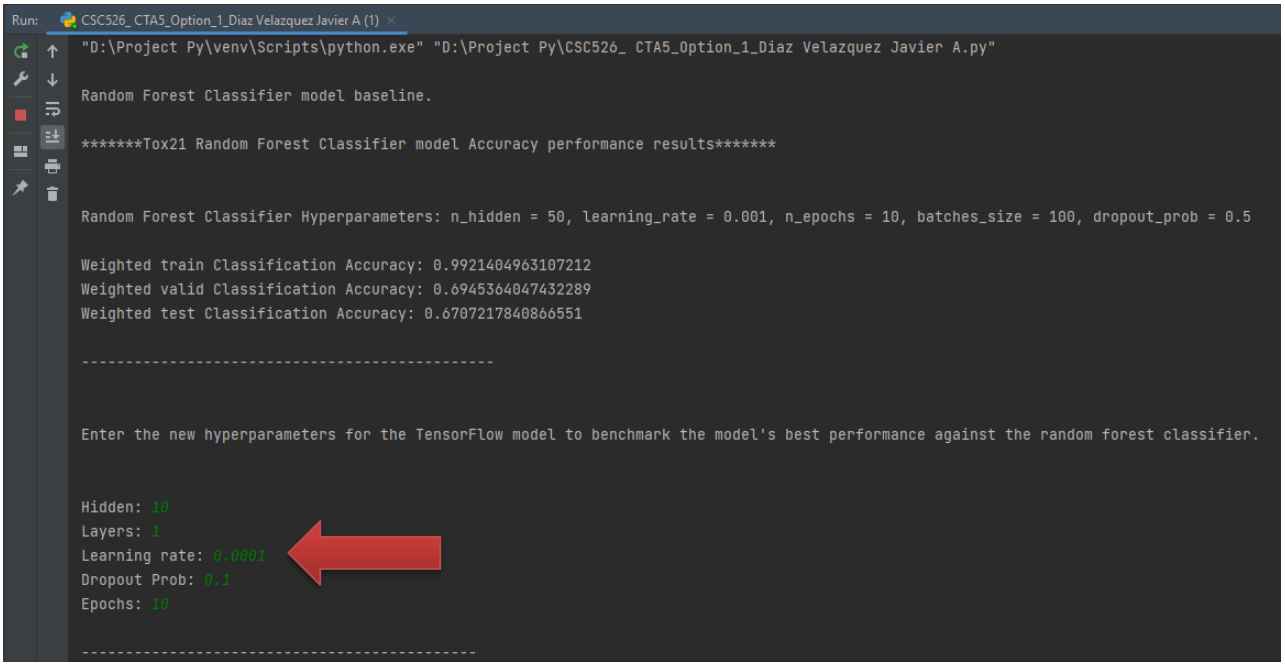

Figure 1 – 3 Hyperparameter settings of the first execution.

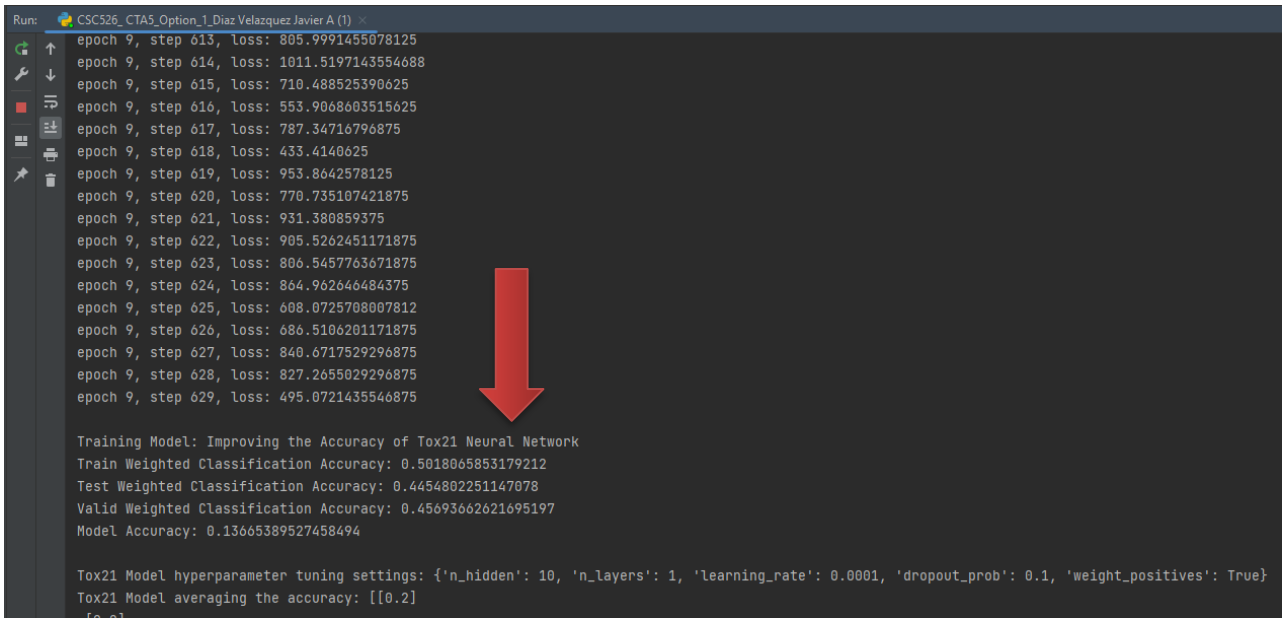

Figure 1 – 4 Tox21 TensorFlow Model first execution result with the list of the hyperparameters.

Best Accuracy Result

Based on the tests and analysis of the model's different hyperparameters setting and results, the best-performing hyperparameter during runtime was the third execution, which contains the following settings 100 hidden, two layers, learning rate at 0.01, dropout at 0.5, and 100 epochs; which yielded an impressive 92% in train-weighted classification accuracy, 64% in testing classification accuracy, a valid accuracy of 58%, and an overall 96% performance accuracy.

Overall, executions two and four also yielded good accuracy results; execution number one was the worst by far, which means that a low hyperparameter setting potentially impacts the model's overall performance.

Conclusion

This exercise successfully demonstrated how deep neural networks are often quite sensitive to different hyperparameter settings and finding the right combination of hyperparameters can significantly impact the performance and training behavior of the network. In addition, due to this sensitivity, it is common practice to perform hyperparameter tuning using techniques like grid search, random search, or more advanced methods like Bayesian optimization. Regular experimentation and tuning are essential to finding the best set of hyperparameters for the problem the deep network intends to resolve. Finally, an Improper combination of hyperparameter settings can lead to issues like slow convergence, poor generalization, or even training instability; therefore, careful experimentation, tuning, and monitoring during training can help improve the performance and stability model.