Check-In #3

Jade Ducharme, Egor Serebriakov, Aditya Singh, Anthony Wong

April 2024

Introduction

Jet tagging is a vital area of research in particle physics, focusing on the process of identifying the fundamental particles initiating a jet—a cone of secondary particles generated when high-energy particles collide. At CERN, the enormous data generated during collisions, several petabytes per second, cannot be fully stored, necessitating real-time, efficient jet-tagging algorithms to determine which collision events to preserve for analysis. With the impending High-Luminosity Large Hadron Collider (HL-LHC) upgrade set for 2029, which will further increase data rates, there is a pressing need for faster, accurate solutions, prompting a shift towards Deep Learning. Our project aims to enhance the speed and maintain the accuracy of a state-of-the-art Neural Network for jet tagging by employing Transfer Learning to translate insights from a high-resolution "teacher" model to a "student" model trained on reduced data, potentially improving real-time data processing capabilities. We plan to leverage publicly available datasets and collaborate with experts like Professor Gouskos to refine our approach and optimize our model architecture.

Challenges

The hardest part of the project we encountered so far has been preprocessing. Indeed, the ATLAS Top Tagging Open Data Set we have been using comes with several features (on the jet and constituent levels, as well as some high-level features). These features, which include the jet (and individual consituent) mass, momentum, energy, pseudo-rapidity, and azimuthal angles, have been difficult to interpret and we had to consult with Prof. Gouskos in order to find out exactly which features are necessary to build our models.

In the end, we found that the most meaningful information is stored at the constituent level. Our final (preprocessed) data therefore has shape `[input_size, num_feature, num_constituents]` = `[input_size, 4, 80]`.

The second hardest part of the project so far has been building the Graph Neural Network. Indeed, not only is this type of network conceptually challenging, it is also difficult to train due to its sensitivity to hyperparameters and the great amount of computational resources required for training. We have a "beta" GNN up and running now, but it performs very poorly and runs very slowly on our local machines.

Insights

We now have the Teacher FCNN up and running, but we are still working on figuring out the best set of hyperparameters and tuning the layers and so on. Currently, it shows obvious signs of overfitting and has overall poor performance, which is somewhat expected given our small training size (10,000 – about the limit of how many items the GNN can run with on our local machines) and the complexity of the data.

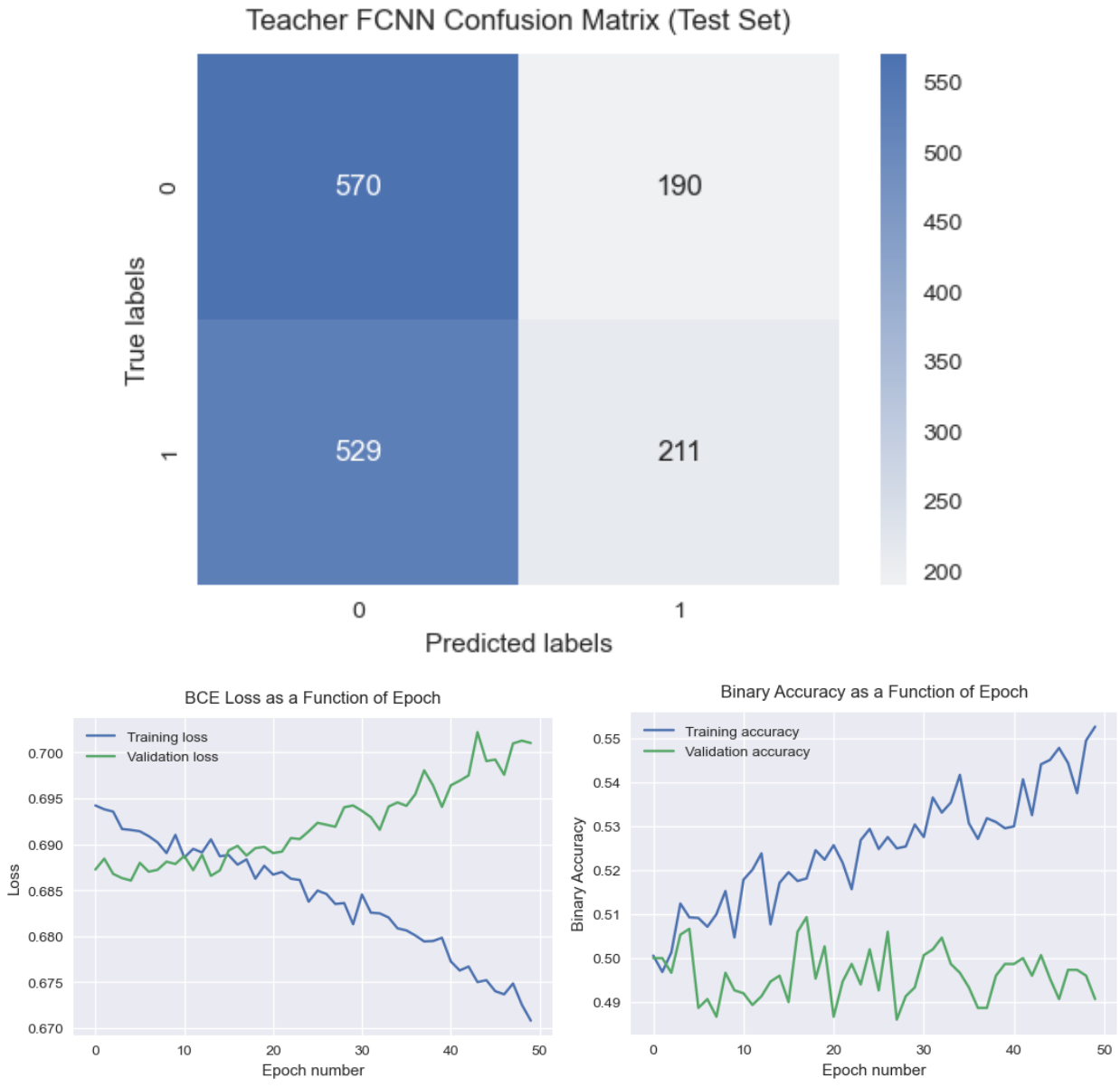

Figure 1: Teacher FCNN results so far

We are actively working on setting up transfer learning with the Student FCNN.

We also have the Teacher GNN up and running, but as mentioned earlier, it performs very poorly and runs extremely slowly on our machines.

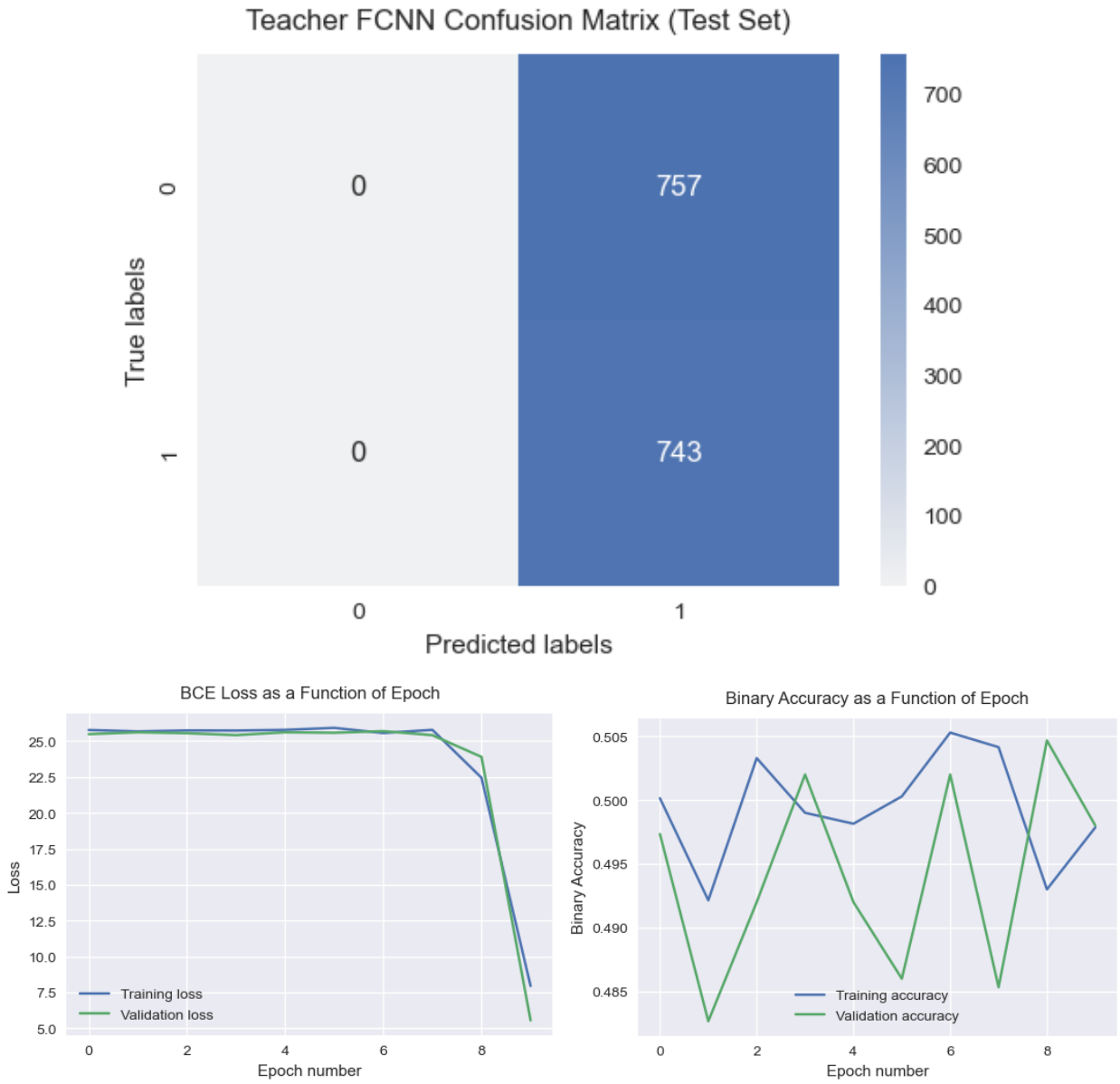

Figure 2: Teacher GNN results so far

Plan

Immediate next steps include switching to Oscar for training the GNN. Indeed, we can use Oscar's GPUs for training, which should provide a significant speedup compared to our local machines' CPUs.

We also need to finalize the Student FCNN and Student GNN, and implement transfer learning. We are almost done with this but are working on getting rid of some final bugs.

We will also most likely need to dedicate quite a bit of time to hyperparameter tuning to try and get the best possible results.

And finally, we would like to meet with Prof. Gouskos at least one more time to get his professional input on how to improve our implementation given his knowledge of the dataset and overall scientific goals of the project.