Draft version May 12, 2024
Typeset using LaTeX default style in AASTeX631

Underlying Event Study: Physics 594

Adam R. Vendrasco[1]

[1] *University of Tennessee*
1408 Circle Dr, Knoxville, TN 37996

ABSTRACT

This study focuses on measuring the underlying event activity in a proton-proton collision at a center-of-mass energy of $\sqrt{s}$= 13 TeV conducted during Run 2 of the Large Hadron Collider at the Compact Muon Solenoid Detector. Applying machine learning techniques, I attempted to predict characteristics of the underlying events to further constrain properties of the Z-boson using a Z $\rightarrow\mu^+\mu^-$ data set. Particularly, I attempt to see if the momentum in the z-direction of the Z-boson can be predicted from non-muon PF Candidates. This data was acquired from the Compact Muon Solenoid Open Data source originally taken in 2016.

1. INTRODUCTION

The Large Hadron Collider (LHC) is the world's largest and highest-energy particle collider which produces proton-proton (pp) beam center-of-mass collision energies at $\sqrt{s}$ = 13 TeV. From the composite nature of protons and at these energy scales, the individual partons can be considered to be the primary interactions within the collision event. Due to the increase in parton densities in the pp collision, there is a significant rise in the probability that more than one parton-parton scattering event can occur in the same pp collision (6). These parton-parton scattering events result in large amounts of particle production which can be mainly described by hadronic jets originating from the parton-parton interaction point. These jets generally have momentum exchanged above several GeV/c and can be referred to as *"hard"* scattering interactions. However, due to the aforementioned parton density increase, there can be several *"softer"* parton-parton interactions known as underlying events (UE) (4). These underlying events are commonly defined as the set of all final-state particles that are not associated with the initial hard-parton scattering and will have a relatively small transverse momentum of a few GeV/c. Since there is a high particle multiplicity after a collision, the UE is often suppressed. This work attempts to utilize machine learning (ML) techniques to further constrain the properties of the Z-boson by incorporating UE. In this study, I specifically attempted to see if it's possible to predict the z-direction momentum (P_z) of the Z-boson by training on UE P_z. If successful, there then is the opportunity to apply this to other properties like the mass of the Z-boson.

2. DATA SET AND EVENT SELECTION

2.1. *CMS Data Processing*

The Compact Muon Solenoid (CMS) is a particle detector stationed at one of the four primary collision points around the LHC. The CMS detector is a complex instrument consisting of several sub-detectors arranged concentrically around the interaction point. These sub-detectors include the Tracker, Electromagnetic Calorimeters (ECAL), Hadron Calorimeters (HCAL), and the Muon System. Data processing starts with reconstructing individual particles based on the signals recorded in these sub-detectors. Charged particles leave tracks in the Tracker, while electromagnetic particles (such as photons and electrons) deposit energy in the ECAL, and hadrons (such as protons, neutrons, and pions) deposit energy in the HCAL. Once the individual particles are reconstructed in each sub-detector, the information is then linked from different sub-detectors to identify and reconstruct each particle's initial state and properties. Then using sophisticated reconstruction algorithms, the reconstructed tracks and energy deposits are combined to form a global event description (6). This process associates tracks in the Tracker with energy deposits in the calorimeters and muon chambers, optimizing the reconstruction of each particle's properties. The reconstructed objects, now known as *Particle Flow* Candidates (PFCandidates), and their properties are stored mainly into a ntuple known as a TTree. The details and structure of a TTree are not necessary, but in this data format, one can access various properties of the PFCandidates.

2.2. *Data Set*

Periodically, the CMS collaboration will release data collected at the detector to the public for further analysis. This encourages accessible open science that people outside the collaboration generally would not have access to. As of April 4th, 2024, the CMS collaboration announced the first release of 13 TeV proton-proton collision data collected in 2016. In total, this collision data has over 70 TB of 13 TeV collision data and 830 TB of corresponding simulations which is available to the public through the CERN Open Data Portal. This data set was released in a format called an "AOD" (Analysis Object Data) file. An AOD houses a comprehensive set of reconstructed physics objects that the CMS detector observed during the event. However, due to the AOD file size and complexity, it can be computationally taxing to process the whole AOD file in an analysis. Therefore, a reduction and compression of AOD files are necessary. The data flow and its reduction can be seen below in figure 1.

Figure 1. Flowchart of the data evolution of the file size. At each stage, the file size is reduced.

Eventually getting to the NanoAOD format, total file sizes by about 95% and storing data in standard structures that can be analyzed without dedicated CMS software (4). This format takes a big step towards easily reusable and accessible CMS data.

The data in this study utilizes a ROOT NanoAOD file. This NanoAOD specifically was selected because it retained all particle track information which carries vital information about the reconstructed particles. It is rare to have track information in a NanoAOD format because tracks take up a large amount of storage information and are usually discarded in the transition from MiniAOD to NanoAOD file reduction (5). The specifics of the data selection process on this data can be found in section 2.3.

2.3. *Event Selection*

Within this NanoAOD, one ROOT file was selected at random which our study ran on. Since I am interested in constraining the Z-boson properties, I would like to select events in which the Z-boson decays promptly to a clean source with a low background. One example of a low background decay mode for a Z-boson is Z $\rightarrow\mu^+\mu^-$ which can be seen in figure 2.

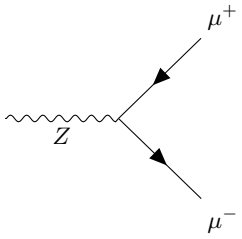

Figure 2. Z $\rightarrow\mu^+\mu^-$ Feynman Diagram

In this randomly selected ROOT file, properties of the observed particles are stored inside a ntuple of which I can access via a data readers like UPROOT. How these properties are physically measured can be found in section 2.1.

For the event selection process, I make cuts to ensure quality data while also limiting how much data is read in to save on computational intensity. I then filter based on the following criteria listed in Table 1. The first cut requires there be exactly two muons in each event. The possible branching decays of the Z-boson are as follows,

$$\begin{aligned} \mathrm{Z} &\rightarrow e^+e^- \\ \mathrm{Z} &\rightarrow \mu^+\mu^- \\ \mathrm{Z} &\rightarrow q\bar{q} \end{aligned}$$

Since the Z-boson is a neutral particle, due to charge conservation, if there are any additional muons present in the PFCandidate list per event, then the observed muons did not originate from a Z-boson decay. The pseudorapidity and

transverse momentum cuts are implemented for data quality purposes. The CMS detector has a range of $|\eta| \approx 4.0$, however, at large $|\eta|$, there is a rise in the particle multiplicity, leading to a lower signal-to-noise ratio (SNR). Similarly, low-momentum particles also contribute to a decreased SNR, as many background particles tend to have low momentum which corresponds to a lower SNR.

Table 1. Data Cuts

Data Requirement	**Description**
Muon Count	Z $\rightarrow\mu\mu$, I require exactly two muons in each event.
Pseudorapidity (η)	$\|\eta\| < 2.5$
Transverse Momentum (pT)	pT > 2 GeV

3. MACHINE LEARNING MODEL

As stated in section 1, in this study, I am investigating whether the Z-boson P_z can be predicted by UE. If the model is successful, UE could be used to predict other properties of the Z-boson; for example, more constraints on the Z-boson mass.

One can evaluate whether or not a model is accurately training and predicting the Z-boson P_z by plotting the known values of the combined dimuon pair P_z against the model-predicted combined dimuon pair P_z. While in the associated NanoAOD ROOT file, there are no true values for the dimuon P_z, one can calculate it using TLorentz Vectors (Tlv). Tlv's are a ROOT-specific vector class that enables a user to input known quantities like mass, energy, position... and calculate a desired value. So using Tlv's I can accurately acquire our true value of the dimuon P_z. I call this true value of the dimuon P_z as the "True Label". In a similar fashion, the model prediction of the dimuon P_z I call the "Predicted Label". One indication that my model is successful would be a linear relationship between the True labels and the Predicted labels. This would indicate that as my dimuon P_z pair increases or decreases, so does my model's prediction of the P_z.

In sections 3.1 and 3.2, a description of the full neural network (NN) model and details on the testing and optimization process will be provided.

3.1. *Full Model*

The current NN model for this study was chosen to be a linear regression, sequential deep neural network (DNN). More specifically, a 4-layer network with 10 nodes in the first layer, 100 nodes in the hidden second, 10 in the hidden third layer and finally a one-node output layer. This model is configured to run on the optimizer "Adam" while utilizing the pre-configured Kearas TensorFlow's loss function "Mean Absolute Error"(MAE). In terms of the other notable hyperparameters, table 2 contains a list of all values in the current iteration of the model. The final results from this study and a discussion of those results are located in sections 4 and 5 respectively.

Table 2. Hyperparameters

Hyperparameters	**Values**
Learning Rate (α)	$\alpha = 1.0 \times 10^{-5}$.
Batch Size	300
Number of Epochs	30

A final iteration of the full model architecture can be seen below.

$$
\begin{aligned}
&tf.keras.Sequential([\\
&tf.keras.layers.Dense(10, activation = relu, input_{shape} = (1,)),\\
&tf.keras.layers.Dense(100, activation = relu),
\end{aligned}
$$

$$
\begin{aligned}
&tf.keras.layers.Dense(10, activation = relu),\\
&tf.keras.layers.Dense(1)\\
&])
\end{aligned}
$$

3.2. *Testing and Optimization*

In this section, I will highlight two of the various models I trained. I will include my overall procedure when testing and optimizing the models along with their outputs. These two models are highlighted specifically because they gave me great insight into both my model performance and the overall direction I should take the architecture of the model. I will provide outputs from all other models I trained along with brief descriptions of their architecture and hyperparameters in section 3.3.

Due to the high amount of events in our sample, in the first initial builds of the DNN, intuition led me to consider both a very wide and deep DNN to accurately train on the data. But before any optimization of the model's architecture could occur, I first fixed our initial state of the model to get reproducible results. While this has been removed in the final code output, I originally fixed our random state to a value of 42.

With the random state fixed, I initially ran a 5-layer TensorFlow Keras Sequential DNN with layers 1-5 having the following architecture,

$$
\begin{aligned}
&tf.keras.Sequential([\\
&tf.keras.layers.Dense(100, activation = relu, input_{shape} = (1,)),\\
&tf.keras.layers.Dense(500, activation = relu),\\
&tf.keras.layers.Dense(500, activation = relu),\\
&tf.keras.layers.Dense(1000, activation = relu),\\
&tf.keras.layers.Dense(1)\\
&])
\end{aligned}
$$

In this test, utilizing the same hyperparameters as in Table 2, but using the "Mean Squared Error" loss function, the resulting output for the model loss, validation loss, and prediction on this architecture is shown in figure 3. From the results displayed in figure 3 of the 100-500-500-1000-1 DNN, the sharp spike downwards in the Training Loss (blue) with the somewhat unchanged Validation Loss (orange) most likely indicates that the model is being overtrained on the very high node count. Also, based on the True vs. Predicted Labels plot this model does not predict the P_z of the dimuon decay products to any basic accuracy. This overtraining led me to then scale down my model's complexity.

Drastically descaling my model's complexity, I then decided to train on a very shallow Neural network consisting of a simple hidden layer of 10 nodes. Which can be seen below.

$$
\begin{aligned}
&tf.keras.Sequential([\\
&tf.keras.layers.Dense(10, activation = relu, input_{shape} = (1,)),\\
&tf.keras.layers.Dense(1)\\
&])
\end{aligned}
$$

In this extremely low-complexity model, whose output can be seen in figure 4, this model underfits the data based on the abnormally high and non-converging Validation Loss. This also produced a non-physical relationship between the True labels and the Predicted Labels which is seen from the horizontal "bands". This indicated to me that there needs to be more complexity in the structure but also to potentially change hyperparameters while also looking into other forms of model manipulation.

3.3. *All other model training*

As stated in section 3.2, this section will provide all other model training with brief descriptions of the DNN framework. None of these are the final DNN structure, but this section highlights other options explored in this study. The hyperparameter space for each of these models was explored extensively as well as the addition of other model structures like Batch Normalization and Dropout layers. Overall this section is organized from highest complexity models to lowest.

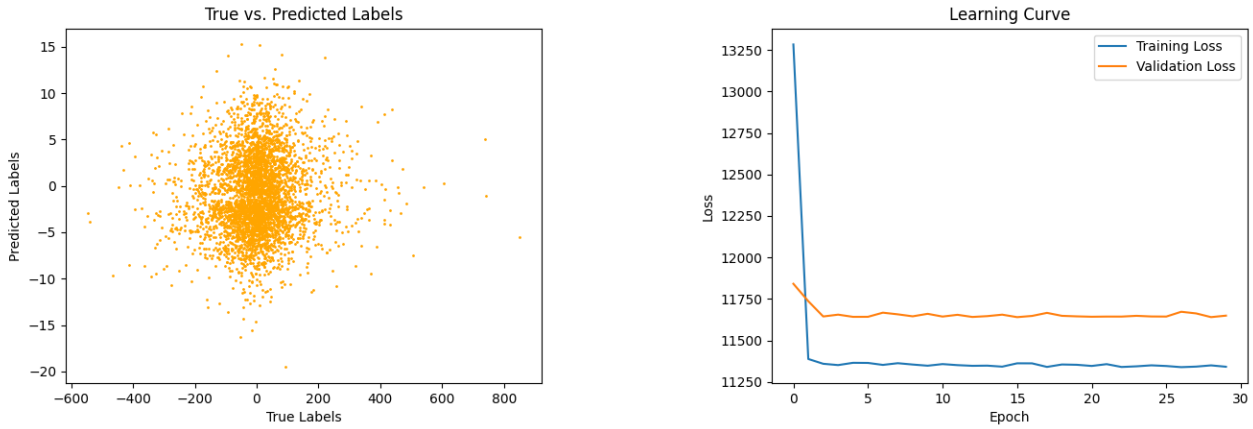

Figure 3. Model prediction evaluation (left) and overall validation and training loss (right) of the 100-500-500-1000-1 DNN architecture. This model also uses the MSE loss function.

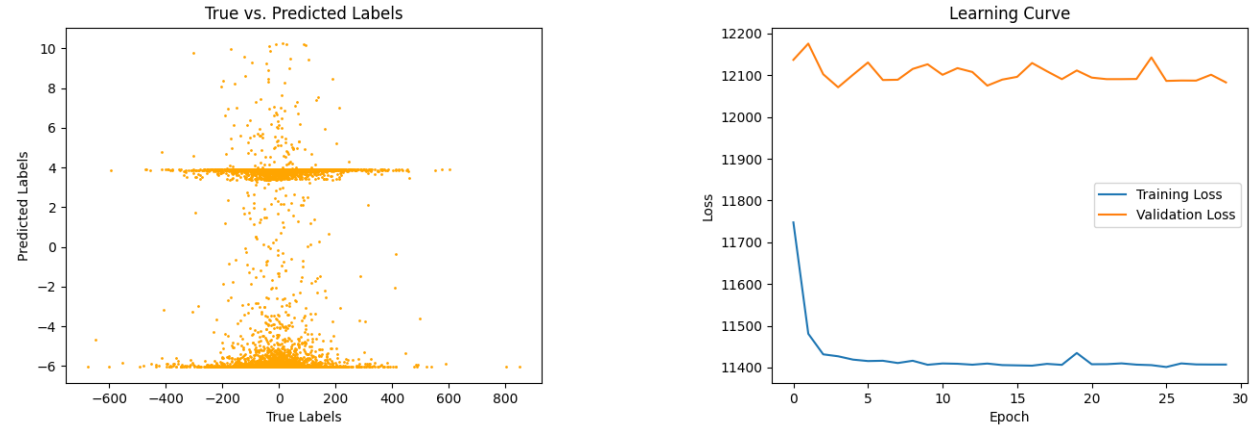

Figure 4. Model prediction evaluation (left) and overall validation and training loss (right) of the 10-1 DNN architecture. This model also uses the MSE loss function.

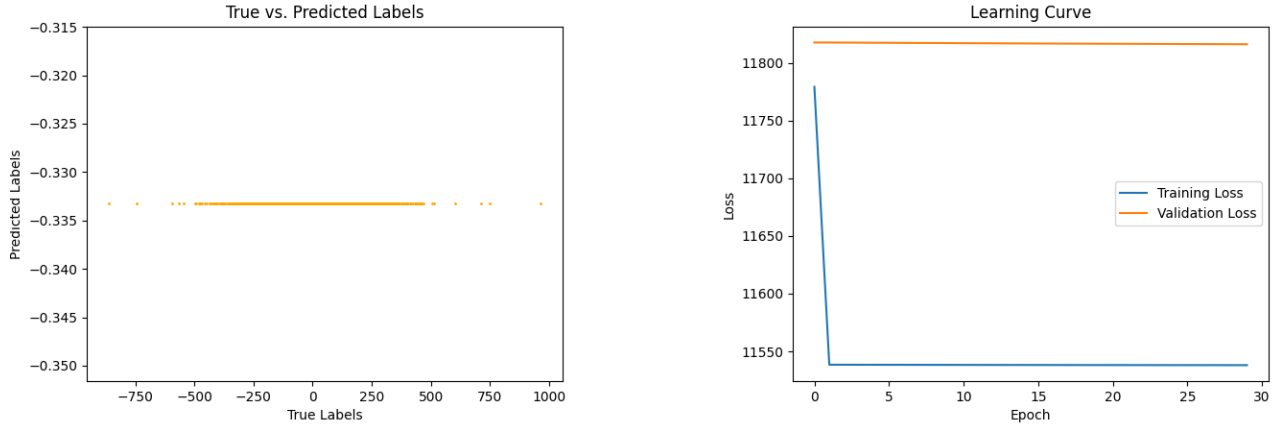

Figure 5. Model prediction evaluation (left) and overall validation and training loss (right) of a 1000-500-500-1000-1 layer DNN architecture and utilizes a standard TensorFlow MSE loss function. Much like figures 3 and 4, this exhibits non-physical attributes in the prediction while also having a non-converging validation loss.

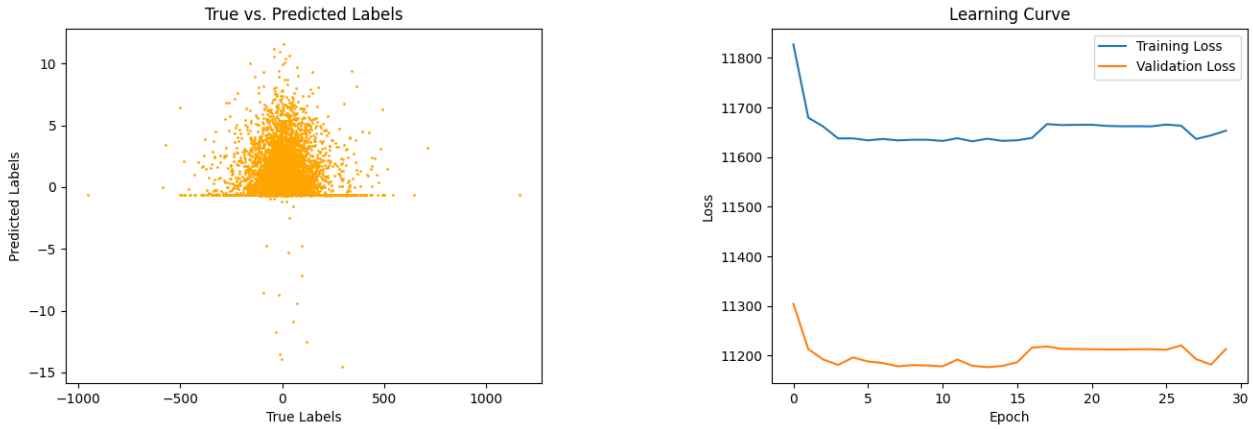

Figure 6. Model prediction evaluation (left) and overall validation and training loss (right) of a 1000-500-130-20-1 layer DNN architecture and utilizes a standard TensorFlow MSE loss function. Also has both non-converging validation and training loss.

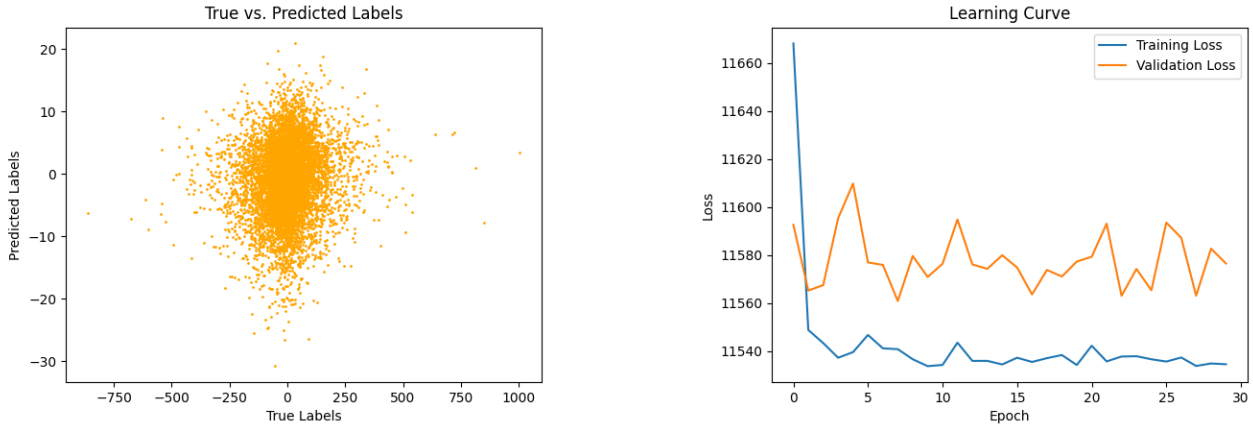

Figure 7. Model prediction evaluation (left) and overall validation and training loss (right) of a 100-500-500-100-20-1 layer DNN architecture and utilizes a standard TensorFlow MSE loss function. Also has both non-converging validation and training loss.

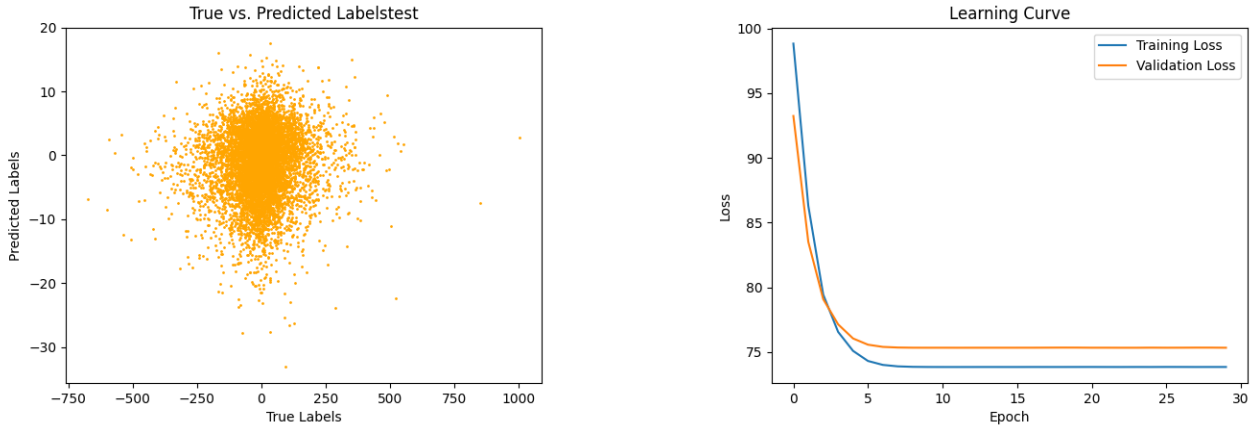

Figure 8. Model prediction evaluation (left) and overall validation and training loss (right) of 10-100-1 layer DNN architecture and utilizes a standard TensorFlow MAE loss function. This has a very good loss for both validation and training.

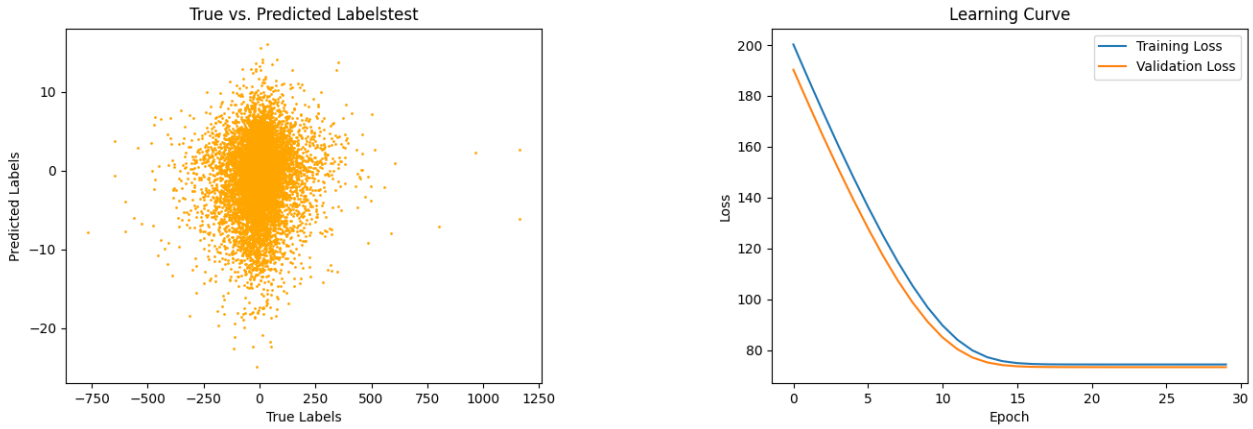

Figure 9. Model prediction evaluation (left) and overall validation and training loss (right) of a shallow 100-1 layer DNN architecture and utilizes a standard TensorFlow MAE loss function. This has a very good loss convergence for both validation and training.

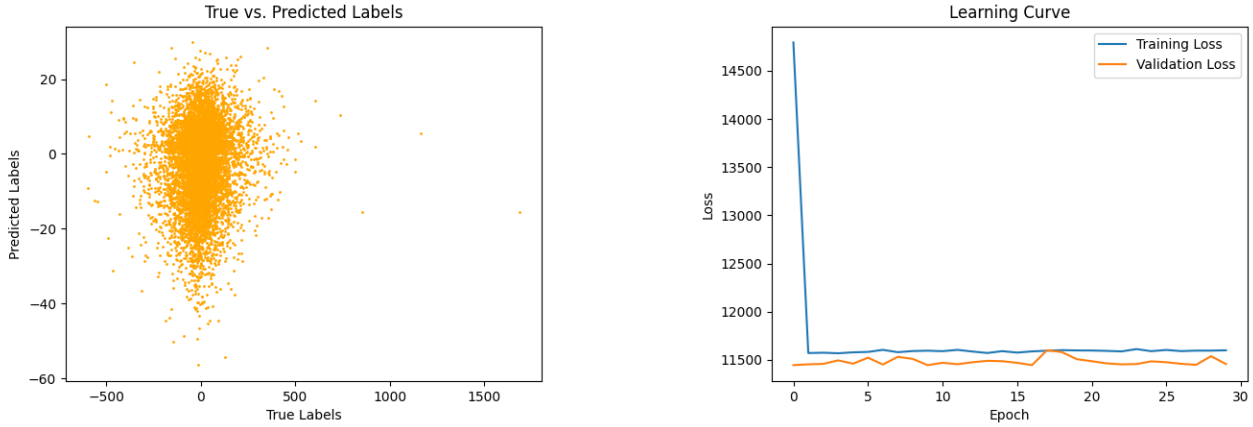

Figure 10. Model prediction evaluation (left) and overall validation and training loss (right) of a shallow 20-1 layer DNN architecture and utilizes a standard TensorFlow MAE loss function.

4. RESULTS

The final model and best network is a 4-layer TensorFlow Keras Sequential DNN with layers 1-4 having the following architecture,

$$
\begin{aligned}
&tf.keras.Sequential([\\
&tf.keras.layers.Dense(10, activation = relu, input_{shape} = (1,)),\\
&tf.keras.layers.Dense(100, activation = relu),\\
&tf.keras.layers.Dense(10, activation = relu),\\
&tf.keras.layers.Dense(1)\\
&])
\end{aligned}
$$

The hyperparameters for this final DNN are identical to the ones listed in Table 2 and the model's training and performance can be seen in figure 11. As seen in figure 11, while there is no distinguishable linear relationship between the predicted and true labels, there is a steeply falling and convergent trend for both training and validation loss. A more detailed discussion of these results can be found in section 5.

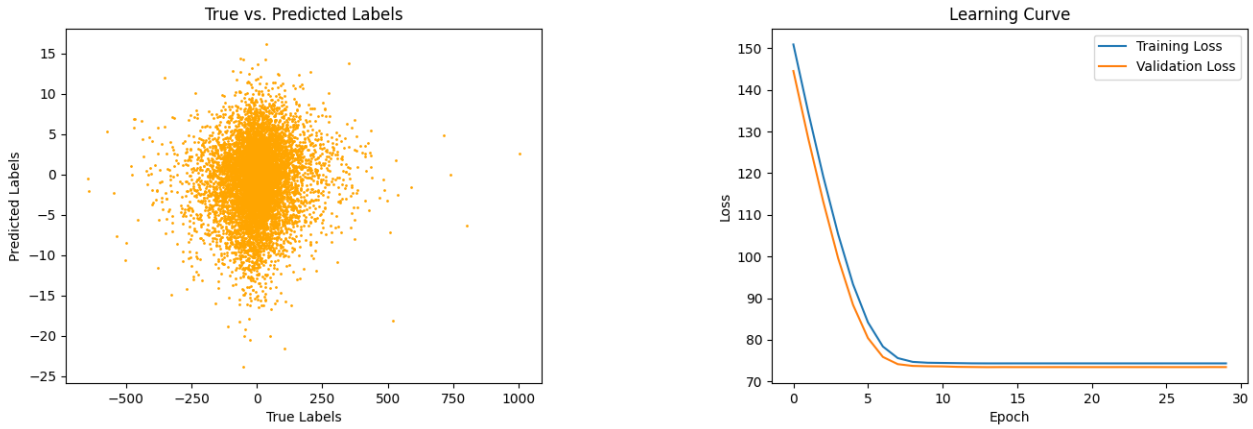

Figure 11. Model prediction evaluation (left) and overall validation and training loss (right) of a 10-100-10-1 layer DNN architecture and utilizes a standard TensorFlow MAE loss function.

5. DISCUSSION

This section aims to investigate the overall model performance as well as interpret the final results displayed in section 4. As you can see in figure 11, the Predicted vs. True Labels distribution does not have any notable linear relationship, which is what is to be expected for accurate predictability of the P_z for the dimuon pair. However, based on the smooth convergent loss for both the training and validation it seems the model is learning well from the test data. Both of these outputs together indicate that the model can train well on the provided training data set but has issues extrapolating outside of the provided data range. This behavior can also be seen in some of the other DNN architecture. The 100-1 and 10-100-1 DNNs, seen in figures 8 and 9 respectively, are especially similar in this regard. So this issue of unpredictability seems to stem outside of model architecture and most likely can be linked to large momentum variability in the muon data sample, as seen by the large range of True Label values. Even when investigating the lower ranges of the True label and Predicted label distribution, there seems to be no coherent linear relationship which can be seen in figure 12. This is unfortunate and more work will need to be done in the future to accurately capture a linear relationship.

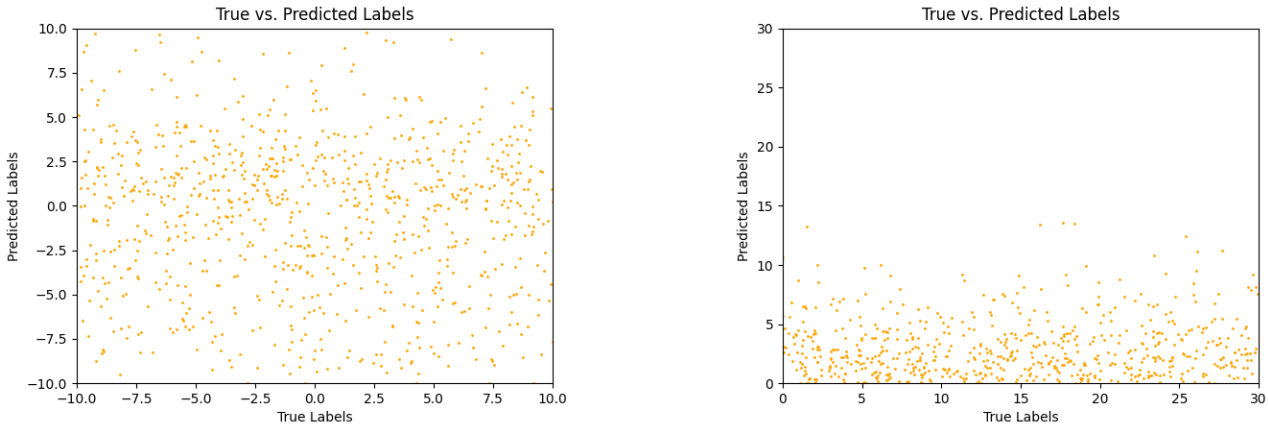

Figure 12. Model prediction evaluation for different ranges to investigate if there is any hidden linear relationships for the 10-100-10-1 model.

5.1. *Next steps and Future work*

As indicated in section 5, the non-linear relationship most likely stems from the large variability in the dimuon pair momentum or the True Labels. I currently plan to present my current model and findings to Dr. Larry Lee who

gave me the idea for this topic. If we wish to continue this work, I believe additional cuts could help in the effort to constrain the dimuon pair momentum. Some cuts could involve primary interaction and vertex cuts. Also, adding and combining additional ROOT files from the 2016 data set could help in this effort.

REFERENCES

[1]G. G. Barnafoldi, A. G. Agocs, and P. Levai. Underlying event studies for lhc energies. In *AIP Conference Proceedings*. AIP, 2011.

[2]J. Butterworth. Underlying events. University College London, 2006.

[3]S. Chatrchyan, V. Khachatryan, and A. M. Sirunyan. Jet and underlying event properties as a function of charged-particle multiplicity in proton–proton collisions at $\sqrt{s} = 7$ TeV. *The European Physical Journal C*, 73(12), Dec. 2013.

[4]C. Collaboration. Cms releases 13 tev proton collision data (2016), 2016. Accessed: May 5, 2024.

[5]I.-H. Collaboration. uproot: Reading and writing root files in python and pure python. IRIS-HEP, 2024.

[6]A. M. Sirunyan, A. Tumasyan, and W. Adam. Measurement of the underlying event activity in inclusive z boson production in proton-proton collisions. *Journal of High Energy Physics*, 2018(7), July 2018.