Individual Report: Machine Learning Taggers for Heavy Electroweak Bosons

Abstract

This report outlines my individual contributions to a group project focused on developing machine learning taggers to distinguish heavy electroweak bosons (W, Z) from Quantum Chromodynamics (QCD) jets. I led the deployment and training of neural network classifiers on a high-performance computing (HPC) cluster, developed the data preparation and labeling pipeline for W, Z, and QCD samples, and designed the methodology for a two-stage classification process. Central to my contribution was the development, training, and testing of the final W versus Z tagger, which was applied to jets previously identified as vector boson-like. I also evaluated the combined classifier performance using ROC and background rejection curves and integrated a mass-based analysis to reinforce our physical insights. In addition, I served as the communications officer, coordinating meetings, scheduling online and physical sessions, and maintaining regular updates with our board member and PhD mentors. My work enabled the successful training and validation of a sequential tagging system capable of suppressing QCD background and differentiating W and Z jets for collider data analyses.

1 Introduction

This project investigated the identification of hadronically decaying electroweak bosons—specifically W and Z jets—against a dominant QCD background using machine learning techniques. Central to the study was the application of LundNet, a graph neural network (GNN) architecture leveraging jet representations in the Lund plane[1, 2].. The classification task was approached in two stages: first, distinguishing W jets from QCD jets while avoiding mass sculpting, and second, differentiating W jets from Z jets within the signal-enriched region.

2 Roles and Context

My project role spanned both technical and enabling responsibilities. On the technical side, I was responsible for managing the computational infrastructure used for training. All machine learning models—specifically the wide-mass W tagger and the second-stage W/Z classifier—were trained on Dias, a GPU-enabled high-performance computing (HPC) cluster maintained by the HEP group. Similar architectures have been evaluated by the ATLAS Collaboration using the Lund Jet Plane for boosted boson tagging [3]. I configured the required computing environment, resolved package and dependency issues, scheduled training jobs using SLURM, and ensured reproducibility via systematic job logging and checkpoint management.

I also developed the data processing pipeline that filtered jets by truth labels, applied kinematic cuts, and transformed the raw ROOT-format data into HDF5 graphs compatible with the LundNet architecture. For the second-stage classifier, I curated jets preselected by the first tagger, relabeled W and Z classes for binary classification, and ensured class balance across training, validation, and testing subsets. Training strategies were refined through iterative tuning of learning rates and batch sizes, and convergence was monitored to prevent overfitting.

As communications officer, I organized meetings, maintained weekly updates, and facilitated internal coordination with mentors and board members. This structured our workflow and ensured group alignment with project goals.

3 Methodology

The training workflow was implemented on Monte Carlo–generated jet data clustered using the Cambridge/Aachen algorithm[4]. The preprocessing pipeline extracted jet substructure features from the Lund plane representation, and I developed a Python script to load, relabel, and filter jets based on classifier scores. For the second-stage W/Z tagger, jets that passed the score threshold from the first-stage W tagger were restructured into a signal-enriched sample. The filtering script, shown in Appendix 1, loaded previously trained model weights, computed classifier scores across batches, and applied a score cut corresponding to 50% signal efficiency.

All training and inference were conducted on the Dias high-performance GPU cluster. Model training jobs were scheduled via SLURM, and resource specifications (e.g., GPU type, memory allocation, job time limit) were defined within a dedicated submission script. This script also handled environment activation, dependency resolution, and logging of standard output files, as shown in Appendix 2.

Classifier performance was quantified using receiver operating characteristic (ROC) curves and background rejection plots, with thresholds chosen to reflect signal operating points of 50% W-jet efficiency. In addition to score-based evaluation, I analyzed the jet mass distribution of the selected sample to cross-check classifier behavior in physical space, as shown in Figure 1. This comparison illustrates how training on different signal compositions affects tagging behavior, including signal peak retention and QCD suppression. Notably, Z jets are partially retained even when not included in training, highlighting the challenge of W/Z separation based on substructure alone.

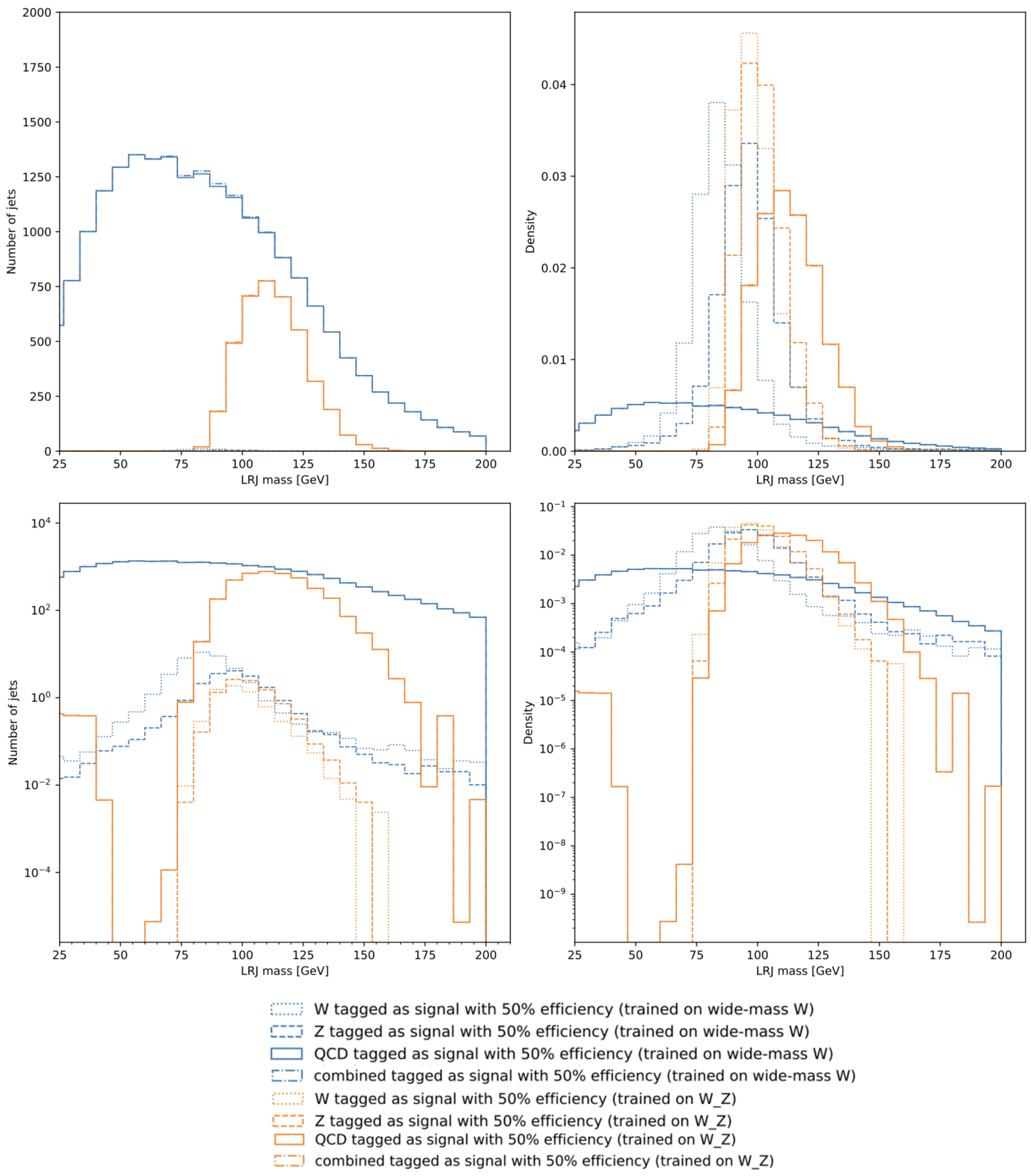

Figure 1: Mass distributions of jets tagged as signal with 50% efficiency, shown for W, Z, QCD, and combined samples. Classifiers were trained either on wide-mass W jets (blue) or on both W and Z jets (orange), and results are shown across four panels using linear/log scales for both counts and densities.

Model behavior and training parameters for the second-stage classifier were defined using a YAML configuration file. This included the graph input paths, batch size, learning rate, number of epochs, and the model checkpoint saving policy. The configuration ensured consistency across training runs and enabled flexibility in architecture selection, with LundNet specified as the model of choice. The file also defined logical links to previously filtered datasets and managed how $\ln k_T$ cuts were applied. The configuration is shown in Appendix 3.

4 Results and Evaluation of Independent Work

4.1 Score Distribution Analysis and Substructure Insights

The initial W tagger score distribution, where W and Z overlap can be seen inFigure 2 , confirming that the tagger was effective for distinguishing boson-like jets from QCD, but not for discriminating between W and Z jets.

The second-stage classifier successfully shifted W jets towards higher scores and Z jets towards lower scores, as shown in Figure 3. While overlap remained, this output indicates that the model learned subtle substructure differences, such as in the splitting scales or radiation patterns encoded in the Lund plane variables[1].

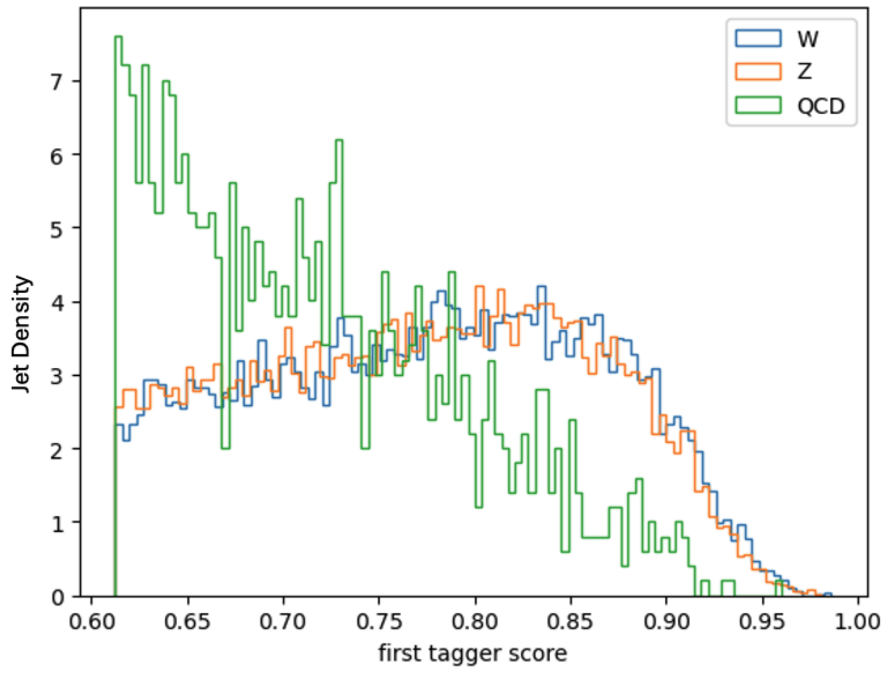

Figure 2: Score distribution from the first-stage wide-mass W tagger. W and Z jets overlap substantially as the tagger was not trained to separate them.

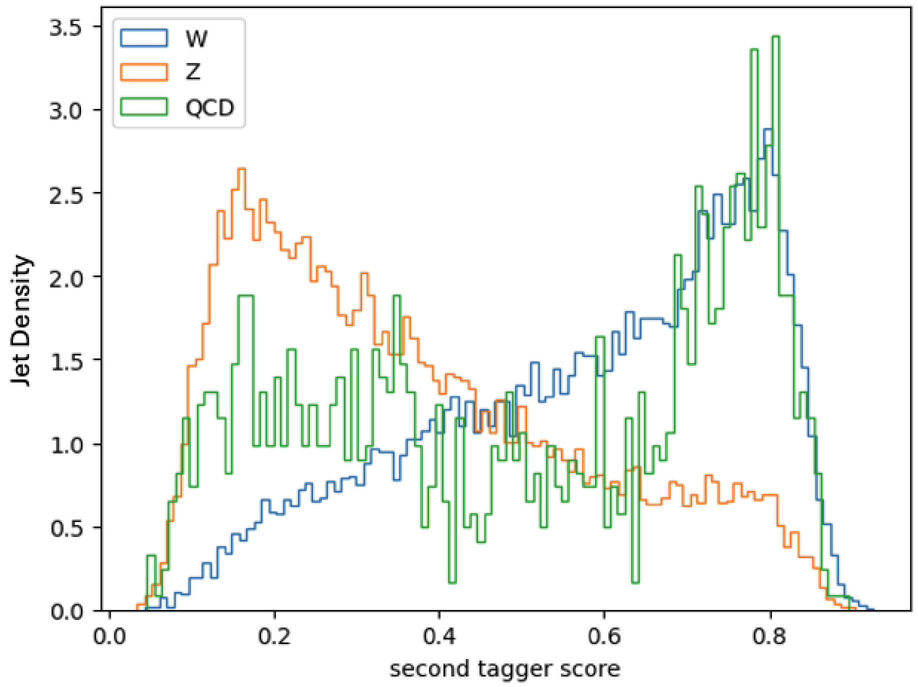

Figure 3: Score distribution from the second-stage W/Z classifier. W jets are shifted to higher scores, indicating improved separation from Z jets.

4.2 Quantitative Evaluation of Classifier Performance

To quantify classifier performance, I evaluated the receiver operating characteristic (ROC) curve (Figure 4) and background rejection factor (Figure 5).

The ROC curve shows that the second-stage classifier outperforms random guessing, with moderate separation between W and Z/QCD jets. At the selected 50% W signal efficiency point, the true positive rate lies between 0.3 and 0.4.

The background rejection plot shows that at this operating point, the classifier achieves a rejection factor just over 10, slightly outperforming a conventional 88 GeV mass cut. This confirms that the classifier captures information beyond mass alone and is learning to identify meaningful substructure[5].

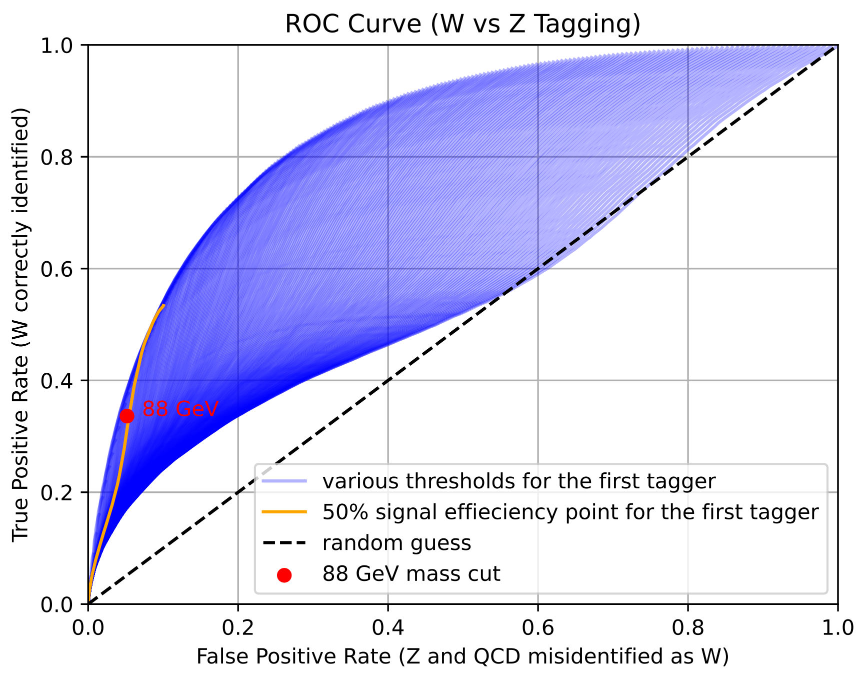

Figure 4: ROC curve for distinguishing W from Z jets using the second-stage classifier. The 50% signal efficiency point and a reference 88 GeV mass cut are marked.

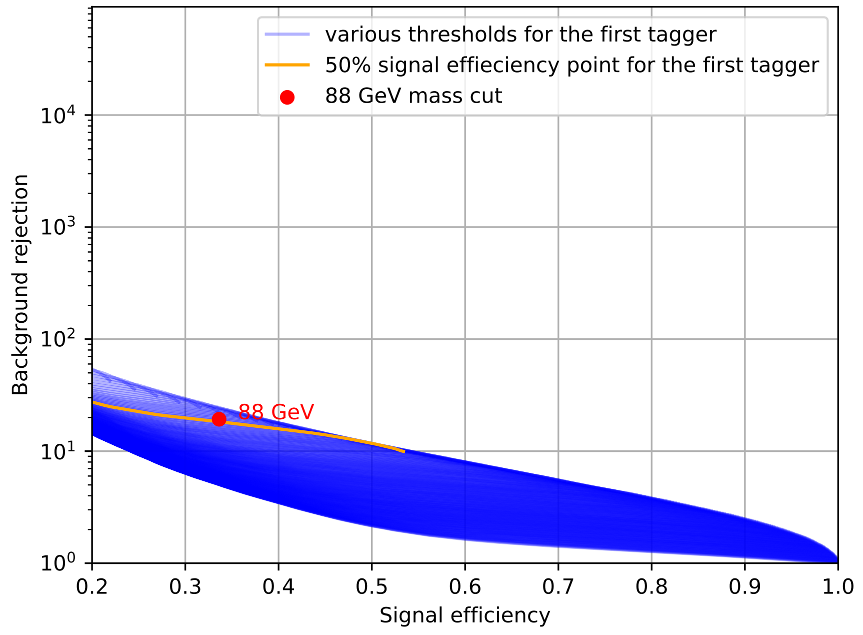

Figure 5: Background rejection versus signal efficiency. The second-stage classifier achieves moderate rejection of Z and QCD jets compared to a mass-based cut.

4.3 Assessment and Limitations

While the two-stage approach improves W jet isolation and suppresses QCD background, full separation between W and Z jets remains challenging. This is consistent with theoretical expectations due to the overlapping substructure and decay topologies of these bosons[6]. The classifier was able to learn useful patterns in the Lund plane variables, but the modest improvement over mass cuts indicates a limit to what can be achieved with current architectures.

Future work could explore deeper neural networks, additional features, or a multi-class classification approach to explicitly model W, Z, and QCD jets simultaneously. Nonetheless, this approach demonstrates a clear improvement over baseline methods and validates the design and training of the second-stage classifier.

5 Reflection and Conclusion

My primary contribution involved designing and implementing a two-stage tagging strategy to distinguish W jets from QCD and Z backgrounds. I developed the second-stage W/Z classifier using LundNet, independently preparing and filtering datasets, configuring YAML-based training setups, and submitting jobs to the Dias HPC cluster. This process required resolving technical challenges such as dependency issues and hyperparameter optimization, alongside applying physics intuition to interpret classifier outputs.

While the classifier yielded only modest gains over traditional mass cuts, it demonstrated the potential of Lund plane–based GNNs in resolving subtle differences in jet substructure[7, 8]. The project thus underscored both the promise and current limitations of machine learning in high-energy physics, highlighting avenues for future improvement using richer input features, deeper networks, or multi-class classification.

In parallel, I acted as communications officer, coordinating meetings, liaising with our board member and mentors, and ensuring regular updates. Balancing this with technical work improved my time management and team collaboration skills. Navigating the iterative and often uncertain nature of ML workflows also deepened my appreciation for the interplay between data-driven techniques and physical insight.

Overall, this experience sharpened both my technical expertise and professional skills, reinforcing my ability to contribute meaningfully to data-driven particle physics research.

References

1. Dreyer, F. A. & Qu, H. *Jet tagging in the Lund plane with graph networks* 2021. arXiv: 2012.08526 [hep-ph]. https://arxiv.org/abs/2012.08526.
2. Dreyer, F. A., Salam, G. P. & Soyez, G. The Lund jet plane. *Journal of High Energy Physics* **2018,** 1–42 (2018).
3. The ATLAS Collaboration. *Tagging boosted W bosons applying machine learning to the Lund Jet Plane* tech. rep. (CERN, Geneva, 2023). https://cds.cern.ch/record/2864131.
4. Dokshitzer, Y. L., Leder, G. D., Moretti, S. & Webber, B. R. Better jet clustering algorithms. *JHEP* **08,** 001. arXiv: hep-ph/9707323 (1997).
5. Alitti, J. *et al.* A measurement of two-jet decays of the W and Z bosons at the CERN p p collider. *Zeitschrift für Physik C Particles and Fields* **49,** 17–28 (1991).
6. CERN. *The Large Hadron Collider* Accessed: 2025-03-06. https://home.web.cern.ch/science/accelerators/large-hadron-collider.
7. Furuichi, A., Lim, S. H. & Nojiri, M. M. Jet classification using high-level features from anatomy of top jets. *Journal of High Energy Physics* **2024,** 1–36 (2024).
8. Larkoski, A. J. QCD analysis of the scale-invariance of jets. *Physical Review D—Particles, Fields, Gravitation, and Cosmology* **86,** 054004 (2012).

A Appendix: Scripts and Training Preparation

A.1 Jet Filtering and Dataset Preparation Script

```
import torch
import numpy as np
from torch_geometric.data import DataLoader
from tools.GNN_model_weight.models import *
from tools.GNN_model_weight.utils_newdata import *

# Load jet datasets
dataset1 = torch.load(".../graphs_W.pt")
dataset2 = torch.load(".../graphs_QCD.pt")
dataset3 = torch.load(".../graphs_Z.pt")

# Load model checkpoint
model = LundNet()
model.load_state_dict(torch.load(".../LundNet_model.pt"))
device = torch.device("cuda")
model.to(device)

# Evaluate on combined W+Z sample
combined_signal = dataset1 + dataset3
loader = DataLoader(combined_signal, batch_size=500)
scores = get_scores(loader, model, device)[:, 0]
truth = np.array([data.y for data in combined_signal])
signal_scores = scores[np.where(truth == 1)[0]]

# Calculate 50% signal efficiency threshold
threshold = np.sort(signal_scores)[-int(0.5 * len(signal_scores))]

# Save filtered datasets
for dataset, name in zip([dataset1, dataset2, dataset3], ["W", "QCD", "Z"]):
    loader = DataLoader(dataset, batch_size=500)
    pred_scores = get_scores(loader, model, device)[:, 0]
    is_signal = pred_scores > threshold
    filtered = [graph for graph, tag in zip(dataset, is_signal) if tag]
    torch.save(filtered, f".../filtered_{name}.pt")
```

Listing 1: Python script used to apply a trained classifier and filter W, Z, and QCD jet datasets based on a threshold corresponding to 50% W signal efficiency.

A.2 SLURM Job Submission Script

```
#!/bin/bash -l
#SBATCH --job-name=tagger_training
#SBATCH -p GPU
#SBATCH -N1
#SBATCH -n1
#SBATCH --gres=gpu:a100:1
#SBATCH --mem=50G
#SBATCH --time=2-00:00:00
#SBATCH --output=/path/to/logs/slurm-%j.out
#SBATCH --mail-user=your.email@ucl.ac.uk
#SBATCH --mail-type=ALL

cd /home/username/GroupProject/lundoptagger
echo "Running␣in:" && pwd

eval "$(/share/apps/anaconda/3-2022.05/bin/conda␣shell.bash␣hook)"
conda activate lundoptagger

echo "Launching␣training␣script..."
python weight_ONLY_TRAINS.py configs/config_ONLY_TRAIN.yaml
```

Listing 2: SLURM batch script used to schedule model training jobs on the Dias GPU cluster, including resource requests and environment setup.

A.3 Training Configuration File

```
data:
  path_to_trainfiles:
    - "data/graphs_filtered_W"
    - "data/graphs_filtered_Z"
  y_values:
    - 0  # W jet label
    - 1  # Z jet label
  path_to_save: "results/models"
  model_name: "results/models/LundNet_best_val_loss_0.150.pt"
  ln_kT_cut: null

architecture:
  batch_size: 2000
  test_size: 0.1
  n_epochs: 35
  learning_rate: 0.0004
  choose_model: LundNet
  save_every_epoch: True

retrain:
  flag: False
  path_to_ckpt: "checkpoints/LundNet_epoch12_val0.124.pt"
```

Listing 3: YAML configuration file specifying model hyperparameters, training inputs, and checkpointing options for the second-stage W vs Z classifier.