Neural-Network Tagger for Hadronic $\boldsymbol{W/Z \to q\bar{q}}$ Jets

Felix Filson

October 13, 2025

1 Introduction

The goal of this project was to build a small neural network (NN) classifier to distinguish hadronically decaying W/Z bosons from generic QCD jets. A tiny Multi-Layer Perceptron (MLP) was trained on Monte Carlo (MC) events generated in Pythia with truth labels, evaluated against a cut-based baseline using ROC and purity curves, and finally applied to ATLAS data to study the large-R jet mass before/after selection.

2 Method

Each event contains at least one large-R jet and one charged lepton (e/μ). For each event, a 10-dimensional feature vector was constructed:

$$x = \big(p_T^j, \eta^j, \phi^j,\ p_T^\ell, \eta^\ell, \phi^\ell,\ \Delta\phi(j,\ell), \Delta R(j,\ell),\ p_T^j/p_T^\ell,\ (p_T^j - p_T^\ell)/(p_T^j + p_T^\ell)\big),$$

where p_T is transverse momentum, η is pseudorapidity, and ϕ is the azimuth. $\Delta R = \sqrt{(\Delta\eta)^2 + (\Delta\phi)^2}$ is the η–ϕ distance between the leading jet and lepton. Mass and energy were omitted from inputs to avoid mass sculpting.

The feature matrix was built for the MC sample together with the truth label $y \in \{0,1\}$ for the leading large-R jet. To avoid information leakage from multiple objects in the same event, the data were split *grouped by event id* into disjoint train/validation/test subsets (60/20/20). All features were standardised using the *training-set* mean and standard deviation; the same scaler was reused when scoring data events.

A small feed-forward MLP was implemented in `PyTorch`:

$$\text{Linear}(10\to32) \to \text{ReLU} \to \text{Dropout}(p{=}0.1) \to \text{Linear}(32\to16) \to \text{ReLU} \to \text{Linear}(16\to1).$$

Training used `BCEWithLogitsLoss` with a `pos_weight` to handle class imbalance. Stochastic Gradient Descent (SGD) plateaued, so Adam was adopted with learning rate 10^{-3} and batch size 64. Early stopping monitored validation loss with patience 10 and restored the best epoch.

To deploy the classifier, a single working point t^* was chosen by *maximising purity* subject to the *signal efficiency* constraint $\varepsilon_S \geq 0.30$. ROC and purity-vs-threshold curves were produced on the MC test split, and the trained model was then applied to data using the frozen scaler and threshold.

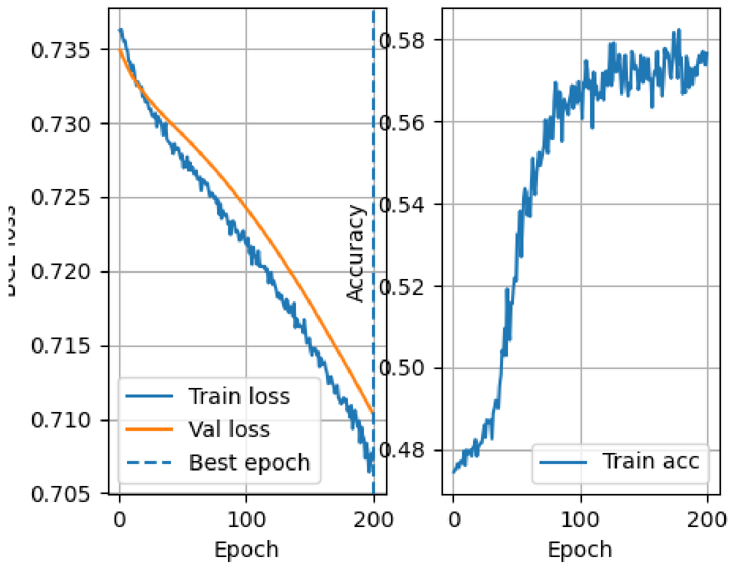


Figure 1: Training history of the MLP using the **SGD** optimiser. The run plateaus and does not converge to a low validation loss within 200 epochs.

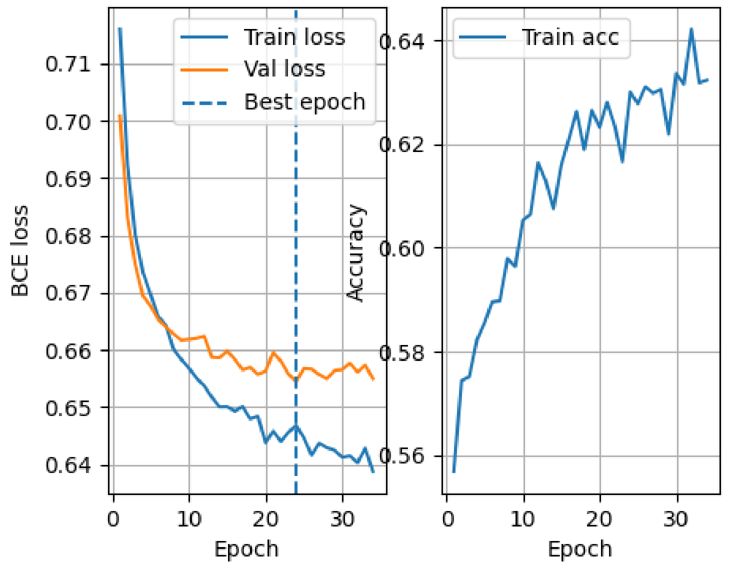


Figure 2: Training history of the MLP using the Adam optimiser. The model converges and early stopping selects the best epoch at approximately epoch 34.

3 Results

3.1 Validation and Test Performance

Figure 1 shows that the **SGD** run plateaus and does not reach a low validation loss within 200 epochs. In contrast, Figure 2 (Adam) shows clear convergence; early stopping selected the best model around epoch 34. For that epoch we obtained:

$$\text{Val loss} = 0.65593 \qquad \text{Val AUC} = 0.668.$$

On the test split the model achieves an **AUC = 0.670**, indicating good generalisation beyond the validation set.

Metric	Baseline (Cuts)	Neural Network ($t^* = 0.640$)
AUC (test)	–	0.670
Purity $S/(S+B)$	0.537	0.626
ε_S (signal efficiency)	–	0.300
ε_B (background efficiency)	–	0.173

Table 1: Performance on the MC test split. At the fixed working point $t^* = 0.640$, the NN improves purity by $\approx 17\%$ over the cut baseline while satisfying $\varepsilon_S \geq 0.30$.

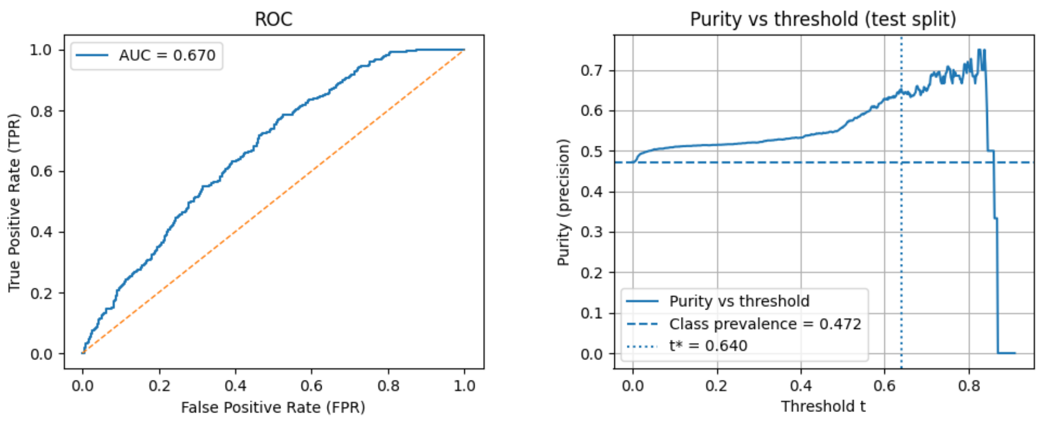
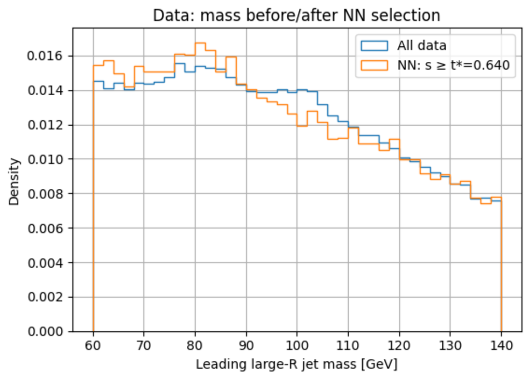


Figure 3: Left: ROC curve on the test split with **AUC = 0.670**. Right: Purity vs. threshold on the test split. The dashed line shows the class prevalence (≈ 0.472); the dotted vertical line marks the chosen working point $\mathbf{t}^* = \mathbf{0.640}$.

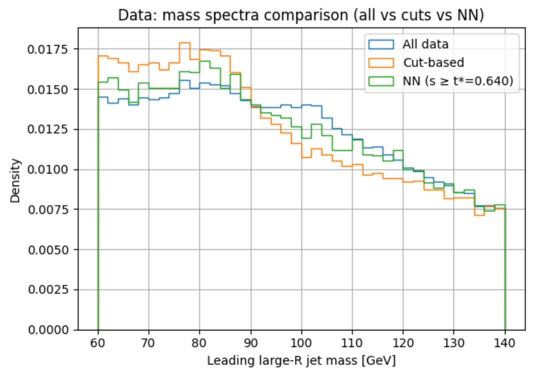
The ROC in Figure 3 (left) summarises global discrimination across all thresholds. To operate the classifier, we fix t^* by maximising purity subject to $\varepsilon_S \geq 0.30$. At $t^* = 0.640$ the NN attains $\varepsilon_S = 0.300$ and $\varepsilon_B = 0.173$, yielding a test-set purity of 0.626, compared to 0.537 for the cut-based baseline (Table 1).

3.2 Application to Data

The trained NN was applied to the data sample (`jets.csv`), using the same feature standardisation derived from the MC training set and the frozen working point $t^* = 0.640$. Figure 4 shows density-normalised large-R jet mass spectra on data. The left panel compares the spectrum before selection with the NN selection $s \geq t^*$; the right panel overlays *all events*, the *cut-based* selection, and $NN(t^*)$. The NN selection visibly enriches the W/Z region (approximately 80–90 GeV) without introducing a narrow artificial peak elsewhere, consistent with the improved purity observed on the MC test split.



(a) Before (all data) vs. NN selection.



(b) All data, cut-based baseline, and NN(t^*) overlaid.

Figure 4: Large-R jet mass spectra on data (density-normalised). The NN(t^*) selection enhances the W/Z mass window ($\sim$ 80–90 GeV) relative to all events and is comparable or better than the cut-based selection, with no evidence of mass sculpting.

4 Conclusions

The neural-network tagger outperforms the simple cut-based selection in both purity and overall discrimination. Using only ten kinematic/angle inputs (no mass or energy), the NN achieves a test AUC of **0.670** and, at the fixed working point $t^* = 0.640$, a purity of **0.626** with $\varepsilon_S = 0.300$ and $\varepsilon_B = 0.173$; the cut baseline yields a purity of **0.537** on the same MC test split. Validation and test AUCs are consistent, and the event-ID grouped split prevents leakage across datasets. On data, density-normalised large-R jet mass spectra show that NN(t^*) enhances the W/Z region without introducing a narrow artificial peak, consistent with higher purity and no evidence of mass sculpting.

Environment and Reproducibility

Python 3.10 with `numpy`, `torch`, `matplotlib`, and `scikit-learn`. A `requirements.txt` is included to reproduce the environment.