Developpement of advanced electron identification algorithms with the CMS detector

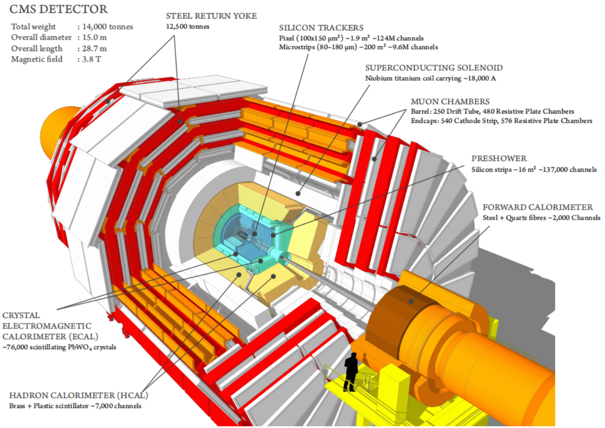
SUN Yuchen

M1 at École polytechnique
Supervisor: Christophe OCHANDO

30/06/2025

Introduction

The Compact Muon Solenoid (CMS) detector at the Large Hadron Collider (LHC) is designed to record a wide range of physics processes. The Inner Tracker reconstructs charged-particle trajectories with a spatial resolution on the order of $10\,\mu$m, while the ECAL measures electron and photon energies with a resolution of better than 5% in the barrel region.



Introduction

The discovery of the Higgs boson in 2012 has opened new avenues for precision measurements of its properties. One of the golden channels is

$$H \to ZZ \to 4\ell,$$

in which four isolated leptons (electrons or muons) are reconstructed. In particular, the $H \to ZZ \to 4e$ final state benefits from the excellent ECAL energy resolution and precise tracking. However, efficient identification of these electrons is challenged by several background sources that reduce the signal signature.

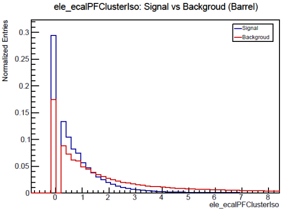
Types of "fake" electrons

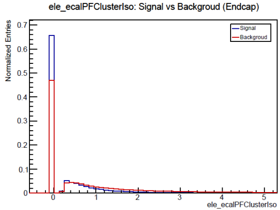
For the $Z \rightarrow e^+e^-$ channel, the principal background sources that can be confused with signal electrons include:

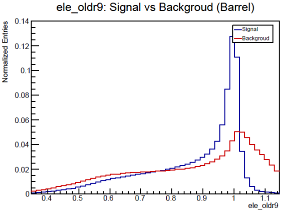
1. Misidentified non-electrons by the reconstruction algorithms, for example charged pions from light-flavor jets, producing electromagnetic showers in the ECAL and being misidentified as electrons.
2. Electrons from heavy-flavour jets, decays of B or C mesons (hadronization of b/c quarks). These arise from displaced vertices and are non-isolated.
3. Electrons from τ decays, which also originate from displaced vertices and non-isolated.
4. Electrons from photon conversions, especially in the endcap regions where additional material before the ECAL increases the conversion probability.

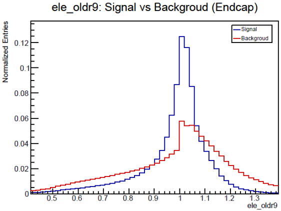
Datasets and Variables

Simulation is based on generating the Drell–Yan + jets process with aMC@NLO; signal electrons originate from Z decay, and background electrons come as described above. For the input variables, there are roughly four categories: Cluster shape, Track information, Track-Cluster matching, and Isolation.









Datasets and Variables

Cluster shape	
ele_sigmaietaieta	standard deviation of η
ele_sigmaiphiiphi	standard deviation of ϕ
ele_scletawidth	width of the supercluster in η
ele_sclphiwidth	width of the supercluster in ϕ
ele_hadronicOverEm	ratio of HCAL energy over ECAL energy around the seed to supercluster (H/E)
ele_circularity	circularity of supercluster
ele_r9	Ratio of the energy in a 3×3 cluster around the seed over supercluster energy
Track information	
ele_kfchi2	χ^2 of the Kalman Filter track fit
ele_gsfchi2	χ^2 of the Gaussian Sum Filter track fit
ele_gsfhits	number of hits associated with the GSF track
ele_kfhits	number of hits associated with the Kalman Filter track
ele_fbrem	fractional momentum loss due to bremsstrahlung ($f_{\mathrm{brem}} = 1 - p_{\mathrm{out}}/p_{\mathrm{in}}$)
ele_nbrem	number of bremsstrahlung photons emitted along the track
ele_expected_inner_hits	number of expected but missing inner-detector hits

Datasets and Variables

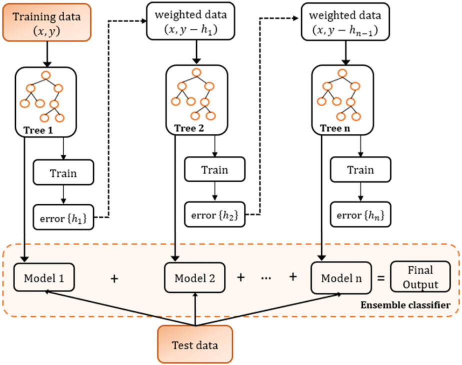
Track-Cluster matching	
ele_ep	ratio of supercluster energy to track momentum at the innermost hit
ele_eelepout	ratio of the closest PF cluster energy to track momentum at the outermost hit
ele_IoEmIop	energy–momentum consistency defined as $1/E_{\mathrm{tot}} - 1/p_{\mathrm{in}}$
ele_deltaetain	absolute difference in η between supercluster center and track extrapolation
ele_deltaphiin	absolute difference in ϕ between supercluster center and track extrapolation
ele_deltaetaseed	absolute difference in η between seed cluster and track extrapolation
Isolation	
ele_pfPhotonIso	sum of p_T of PF photons within cone
ele_pfChargedHadIso	sum of p_T of charged PF hadrons within cone
ele_pfNeutralHadIso	sum of p_T of neutral PF hadrons within cone
ele_pfSumPUIso	pileup-corrected additional p_T sum within cone

Current identification algorithms in CMS

The most direct way is to select a single variable and apply a cut, for example requiring fbrem to exceed a certain threshold. Although the idea of rectangular cuts can be extended to multiple variables and each cut value optimized, this method still makes only a single decision per variable and is therefore not optimal. Another method currently in use is BDT. From a machine learning perspective, the task of distinguishing two processes or categories (in this case electron signal versus background) is a supervised classification problem. Given a set of N observables, the goal is to find the optimal separating boundary in N-dimensional space, and BDT is particularly well-suited for this task.

Current identification algorithms in CMS

BDT iteratively builds a series of weak decision trees, each new tree fitting the misclassified events of its predecessor, then combines all trees' weighted outputs into a strong classifier. Common implementations include Gradient Boosted Decision Trees (GBDT) and other boosting variants.



First step: TMVA

For a quick comparison of the methods, first training was carried out within the TMVA framework. Based on the ROC curves, the currently optimal algorithm is GBDT, although the other algorithms show little difference.

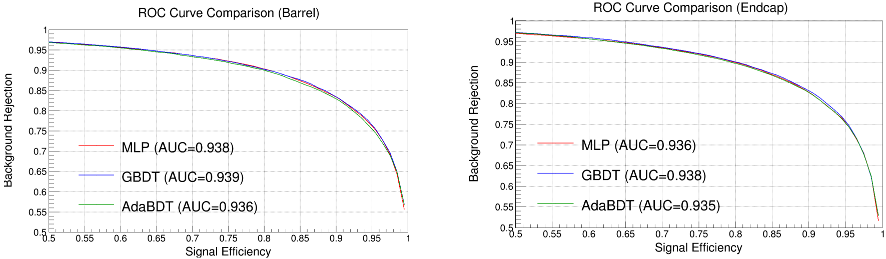
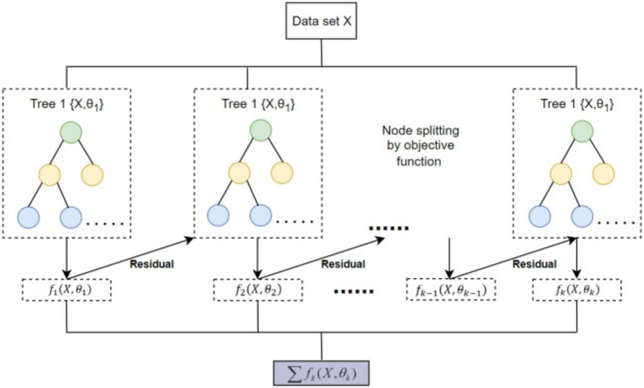


Figure 2: Comparison of ROC curves of various algorithms in the TMVA framework

For the BDTs, a grid search was used to tune hyperparameters. Due to computational constraints, the MLP architecture was determined by manually testing several network sizes.

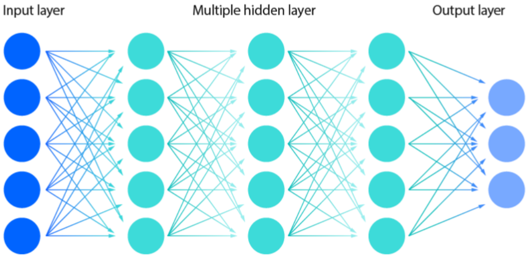
XGBoost and DNN

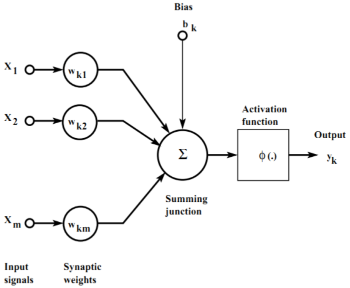
Training with TMVA is constrained, so the effort shifted to a more flexible platform and more modern methods.
XGBoost, as a more powerful BDT model, is an optimized gradient-boosting library that employs second-order Taylor approximation for loss minimization, includes regularization to control overfitting, and supports parallel tree construction and efficient handling of sparse data.



XGBoost and DNN

For neural network training, the PyTorch platform is used, allowing flexible adjustment of network architecture, activation functions, and learning-rate scheduling schemes, which improves the network's performance.





XGBoost and DNN

The training of XGBoost is based on the python package "xgboost".
Regarding the neural network structure: hidden layers are of 512 $\rightarrow$ 256 $\rightarrow$ 128 $\rightarrow$ 64 $\rightarrow$ 32 neurons; each layer followed by BatchNorm1d. Activation function is LeakyReLU. Several different activation functions are tested and found that the network's performance changed little. Considering computational convenience, LeakyReLU remains the best choice.
For learning-rate scheduling, a two-stage schedule was used: In the beginning of training, starting with a learning rate much smaller than "initial" learning rate and then increase it over a few iterations or epochs until it reaches that "initial" learning rate. after warmup, the learning rate decays smoothly along a cosine curve. .

XGBoost and DNN

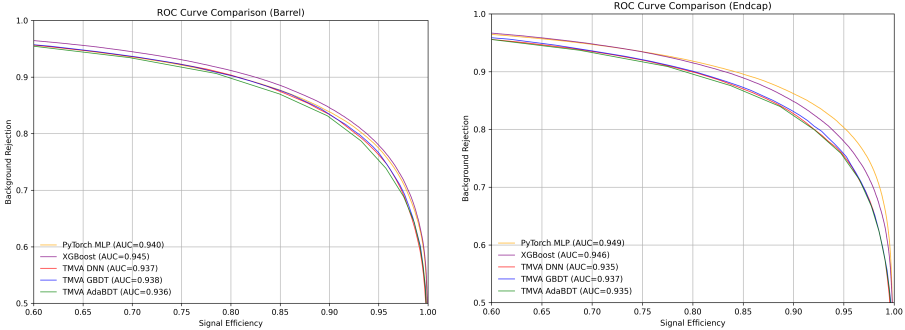


Figure 4: Comparison of ROC curves of various algorithms, including pytorch and XGBoost

Involving eta and pT

In the previous case, we only made a rough division into endcap and barrel regions. Next, we introduce η and p_T and adopt two methods.
First, we partition by η and p_T and train separately, with partitions defined by

$$|\eta| < 0.8, \quad 0.8 \leq |\eta| < 1.479, \quad |\eta| \geq 1.479,$$

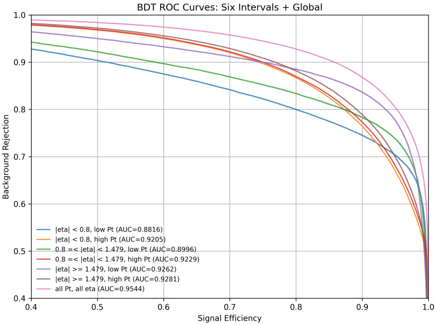
and

$$p_T < 10 \text{ GeV}, \quad p_T \geq 10 \text{ GeV},$$

resulting in six regions.
Second, we use η and p_T directly as input variables.

Involving eta and pT

Neural network structure is $64 \to 128 \to 128 \to 128 \to 64$, $64 \to 128 \to 256 \to 128 \to 64$ or $128 \to 256 \to 512 \to 256 \to 128$, depending on the datasets.



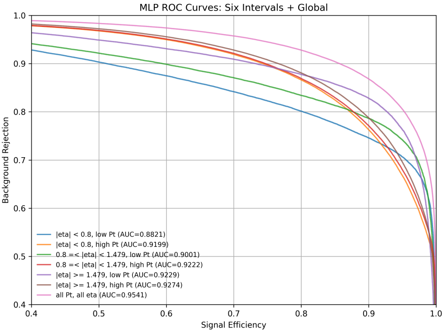
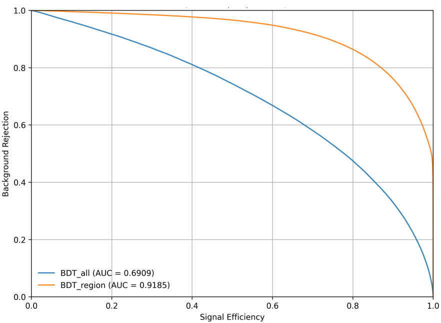
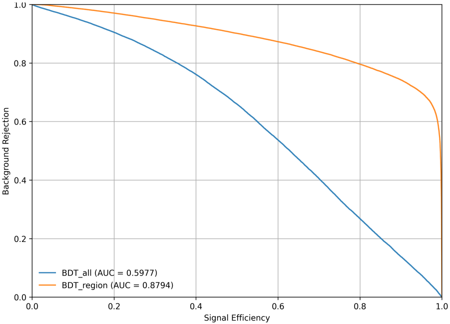


Figure 5: Comparison of ROC curves of all η and p_T regions

Involving eta and pT



(a) $|\eta| < 0.8$ and $p_T \geq 10$

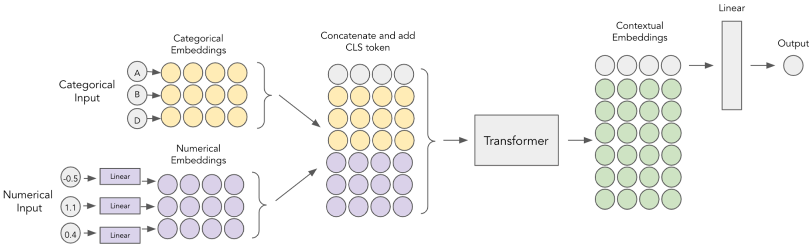


(b) $|\eta| < 0.8$ and $p_T < 10$

Figure 6: Comparison of ROC curves in specific η and p_T regions

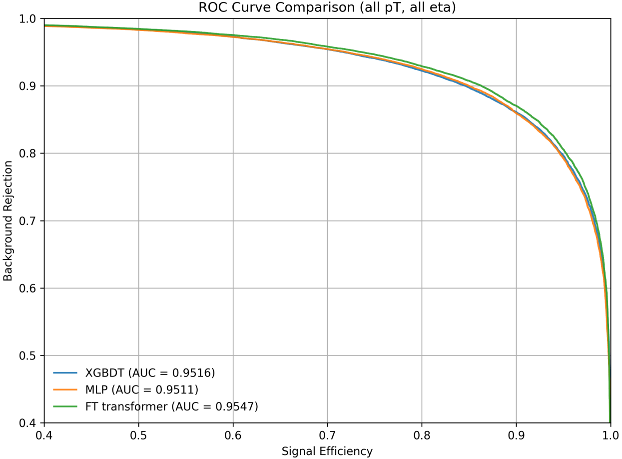
FT-transformer

FT-Transformer is a Transformer model for tabular data, roughly, this model "categorical embendings" to change categorical inputs to tokens and using learnable some linear transformations to change numerical inputs to tokens, thus all inputs have appropriate form for Transformer. And CLS token is another learnable vector added into the tokens. Then put them into Transformer and to get the prediction.



FT-transformer

Under one situation using a small dataset, the FT-Transformer was tested and outperformed both the MLP and XGBoost. The model demonstrated potential to surpass the others, but owing to the heavy computational cost of training and validation, a comprehensive training has not yet been carried out.



Conclusion for Now

According to current conclusions, although the performance of the MLP and other neural networks is not inferior to—and in some respects even superior to—XGBoost, the differences between them are very small. In terms of computational cost for training and validation, XGBoost runs the fastest, and from a practical standpoint, it may still be the optimal algorithm in the current context.

Future Work

Multiclass BDT/NN. Dedicated BDTs are now trained separately for each background situation; in every case, these specialized models outperform the jointly trained network. An intelligent scheduling scheme is being explored to combine specialized classifiers for improved overall performance.

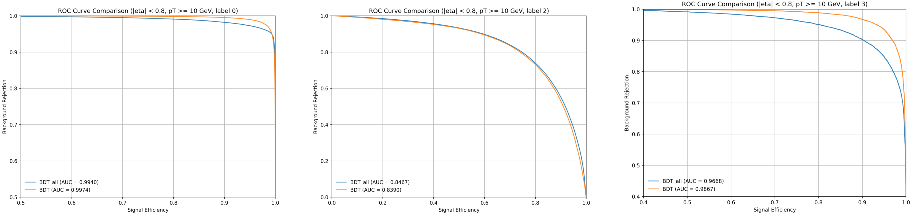


Figure 8: Comparison of ROC curves of specialized models and full model

Future Work

Introduce additional variables: Distinction based on the current variables has reached its limits, and specialized classifiers perform poorly against the second background category (i.e., originating from heavy-flavour jets or τ). Variables matching reconstructed electrons to reconstructed jets—such as the associated b-tagging algorithm score—could strengthen this weak area.

Exploit low-level information: For neural networks—especially CNNs and transformers—the chief advantage is automated feature engineering; by using low-level variables (ECAL reconstructed hit clusters, PF clustering separation, etc.), these architectures may extract previously untapped information and further enhance identification performance.