MODELING Z BOSON DECAY EVENTS WITH GENERATIVE MODELS

Crespi Riccardo - Guidotti Sofia - Picciano Alisia - Reali Alessandro - Toscano Teodoro

June 2025

Abstract

The Z boson is a neutral elementary particle which mediates for the weak force—one of the four fundamental forces that govern all matter in the universe. A Z boson, when excited, decays into two particles: either a muon and an antimuon ($Z \to \mu^+\mu^-$ events in the dataset), or an electron and a positron ($Z \to e^+e^-$ events). An accurate modeling of Z boson decays allows one to test the Standard Model of particle physics, as well as explore beyond current knowledge. While conventional methods are computationally expensive, in this study we investigate the viability of deep generative models—in particular, VAEs and WGANs—as efficient alternatives for simulating Z boson decays, while also training an MLP to distinguish between the two decays. Our findings reveal that VAEs provide interpretable latent representations, while WGANs produce sharper and more realistic synthetic samples. This work demonstrates that generative modeling can serve as a powerful tool for simulation, data augmentation, and unsupervised analysis in high-energy physics.
Link to the dataset: https://www.kaggle.com/datasets/omidbaghchehsaraei/identification-of-two-modes-of-z-boson/data

1 Introduction

The Z boson is a massive neutral particle which is commonly produced at the Large Hadron Collider at CERN via expensive Drell-Yan processes, and decays into fermion-antifermion pairs (electrons and muons). Analyzing the data from these experiments is crucial for Standard Model validation and searches for new physical models. Traditionally, such data are synthetically generated via accurate Monte Carlo simulations, which, despite being accurate, require high computational costs and involve complex modeling steps. Generative models, such as VAEs and WGANs, offer a faster, data-driven alternative. Importantly, they can generate samples that are both statistically significant and physically realistic (e.g., satisfy the invariant mass constraint).

2 Exploratory Data Analysis

To ensure optimal learning and quality in data generation, several preprocessing steps were applied to the data set at hand:

- Removal of irrelevant columns (arbitrary identifiers, specific for each experiment);
- Logarithmic transformation of highly skewed features (particle's momenta);
- Outlier removal via z-score method;
- Standardization of numeric features via `StandardScaler`, except for charge variables, to keep their discrete state space.

3 Unsupervised Learning

As part of the unsupervised learning analysis, we applied several dimensionality reduction techniques, which are Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP), to visualize the high-dimensional feature space of Z boson decay events in 2D and 3D. The goal was to explore whether the different decay channels ($Z \to \mu^+\mu^-$ vs $Z \to e^+e^-$) already formed distinguishable clusters without the use of class labels.

However, despite the application of several different techniques in both 2D and 3D spaces, the visualization revealed that the classes were not linearly separable in the reduced spaces, as it can be seen in figure 1. This suggested us that the classes were not easily distinguishable based only on spatial distributions, neither when using linear projections like PCA, nor when using non-linear ones like t-SNE and UMAP.

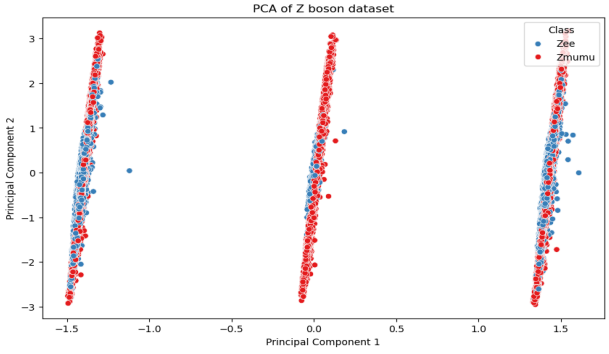

Figure 1: 2D spectral cluseting over the filtered dataset

Despite it all, this step provided valuable insight into the structure of the data, confirming that class separation required more complex approaches.

As a next step, we applied both Spectral Clustering and Hierarchical Clustering to the data, in order to investigate potential natural groupings in the data. This methods aim is to group data points based on intrinsic similarities, without using class labels. However, in both cases, the resulting clusters showed significant overlap, an example is showed in figure 2, and did not align well with the actual decay channels ($Z \to \mu^+\mu^-$ and $Z \to e^+e^-$). This was a further confirmation that the discriminative features separating the classes are not captured effectively by unsupervised similarity-based criteria. This moreover was nonetheless informative as it reinforced the need or supervised approaches for accurate classification in this dataset.

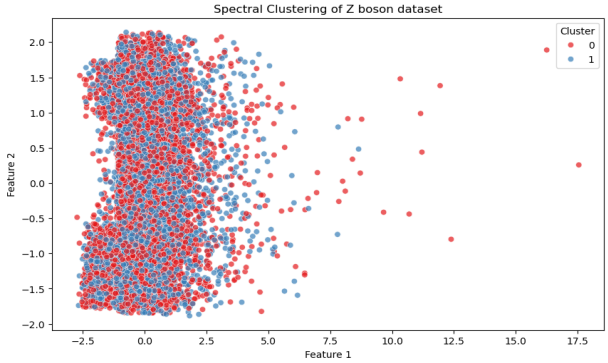

Figure 2: 2D spectral cluseting over the filtered dataset

In particular, for what regards the second method, we added the *Circular Mean Squared Loss*

$$\text{Circular MSE} = \frac{1}{N}\sum_{i=1}^{N}\left(d(\phi_{\text{recon},i}, \phi_{\text{real},i})\right)^2,$$

where

$$d(\phi_{\text{recon}}, \phi_{\text{real}}) = \min\left(|\phi_{\text{recon}} - \phi_{\text{real}}|, 2\pi - |\phi_{\text{recon}} - \phi_{\text{real}}|\right)$$

represents the *circular distance* between ϕ_{real} and ϕ_{recon}. However, despite the theoretical promises of such approaches, neither of them helped improving the accuracy of the model, suggesting an inherent inefficiency of the VAE architecture at capturing periodic quantities, as cen be clearly seen in figure 4.

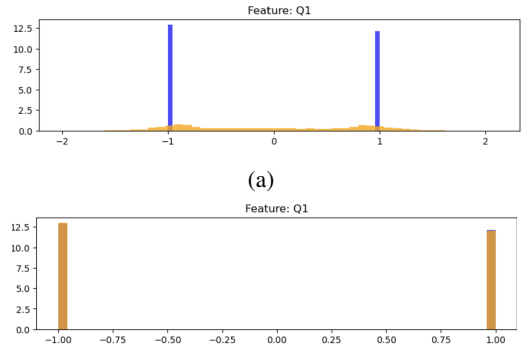

Figure 3: Results of `sign` mapping

4 Variational Autoencoders (VAEs)

VAEs are probabilistic generative models that learn a distribution over a continuous space of high-dimensional data. We apply VAEs separately on the data for $Z \to \mu^+\mu^-$ decays and $Z \to e^+e^-$ decays. After training the model and optimizing its hyperparameters, we noticed that some features of the synthetic data were far from their true distribution. In particular, two different characteristics were poorly modeled: `Q1`, `Q2`, representing the charge of the leptons produced in decay; and `phi1`, `phi2`, measuring some angular dimension. The former issue was efficiently solved by mapping the continuous distribution produced by the VAE into $\{-1, +1\}$ via the `sign` function, as clearly shown in figure 3. For what regards the latter, we addressed it in two different ways:

1. by decomposing the `phi` angle in the training data into its sinus and cosinus components, so as to have continuous distributions;

2. by inserting into the loss function a term accounting for the periodicity of this features.

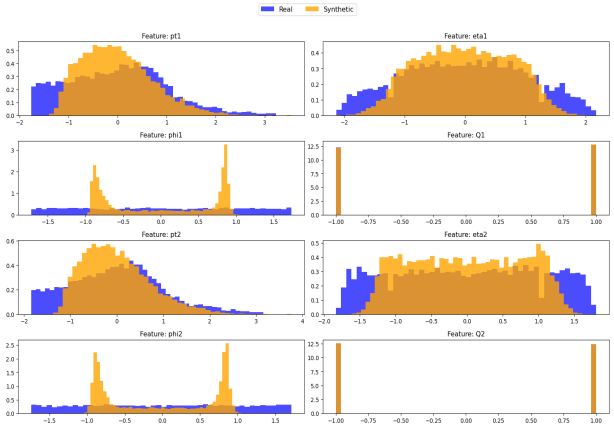

Figure 4: Distribution of Data Genereted via VAE

5 Wasserstein Generative Adversarial Networks

Unlike traditional GANs that minimize the Jensen-Shannon divergence, WGANs optimize the Earth Mover's

(Wasserstein-1) distance:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma}[\|x-y\|],$$

which provides a smoother and more informative loss landscape. The discriminator is replaced by a critic D, which is trained to approximate the Wasserstein distance. To ensure that D is 1-Lipschitz continuous, weight clipping is applied:

$$\theta \leftarrow \text{clip}(\theta, -c, c).$$

WGAN models were trained on the preprocessed Z boson decay data sets following these steps:

1. Training a first Wassertein-based GAN and asses its performance;
2. Upgrading to a WGAN with a gradient penalty, achieving higher performance;
3. Tuning hyperparamters of our best model so far;
4. Givnig a shot at a relatively new GAN implementation (based on a specific energy function); however, despite its appealing physics-based implementation, it led to unsatisfying results.

Overall, the best model turn out to be a GP-WGAN suitably optimized over its hyperparameter space. Indeed, not only is it able to produce synthetic data with a very low Wasserstein distance (≈ 0.04), but also the physics-specific constraints are perfectly met by the generated data points (see figure 5, proving both statistically low error and subject-specific consistency of our model.

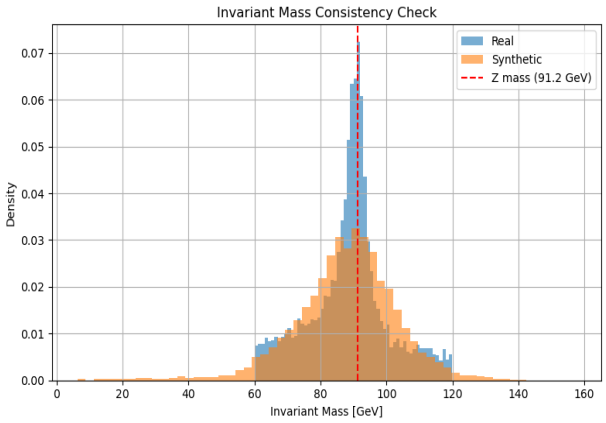

Figure 5

6 Classification task using MLP classifier

In order to investigate the classification task we trained an MLP; in particular, after filtering the dataset from the outliers, we split it in training set, validation set and test set. In order to train the Multi Layer Perceptron, we used the MLPClassifier() built in class in scikit learn library where it is possible to inspect different combination of parameters to find best classification model and implement several regularization method. In particular, as hyper-parameters we inspected among:

- activation: relu, tanh, logistic;
- solver: adam, sgd;
- learning rate: loguniform distribution over (1e-5, 1e-2);
- batch size: 32, 64, 128, 256;
- size of hidden layers: (50,), (100,), (200,), (50, 50), (100, 50), (200, 100),(100, 100), (150, 100, 50)

and for regularization techniques we inspected:

- Strength of the L2 regularization: loguniform distribution (1e-6, 1e-2)
- early stopping: True.

For data augmentation and dropout regularization we consider a different section below. Moreover, dependent on the type of solver (Adam or SGD) we may have different types of further hyper-parameters. In particular, for Adam we may have

- Exponential decay rate for estimates of first moment vector in adam: uniform distribution over (0.8, 0.199);
- Exponential decay rate for estimates of second moment vector in adam uniform distribution over (0.9, 0.0999)

while for SGD we have

- type of learning rate: constant, invscaling, adaptive;
- momentum for update: uniform distribution over m (0.8, 0.199);
- exponent for inverse scaling: uniform distribution over (0.1, 0.8)

while, for type of momentum type and for warm_start boolean parameter we left the default Nesterov and False ones. In order to find the best combination of hyper-parameters we used the RandomizedSearchCV() class included in the sklearn.model_selection library with parameters n_iter=100, cv=5.

The output best parameters have been

- activation: logistic;

- solver: adam;
- learning rate: 0.00851;
- batch size: 32;
- size of hidden layers: (200,)
- strength of the L2 regularization: 9.7497 e-5;
- early stopping: True.
- Exponential decay rate for estimates of first moment vector in adam: 0.82947;
- Exponential decay rate for estimates of second moment vector in adam: 0.99967.

This first version of the model gave a training score of 0.8819, validation score 0.8769 and test score of 0.8803 and a ROC curve with area 0.95. In particular, the confusion matrix and the ROC curve, represented relatively in figures 6 and 7, highlight a good classification of the two classes.

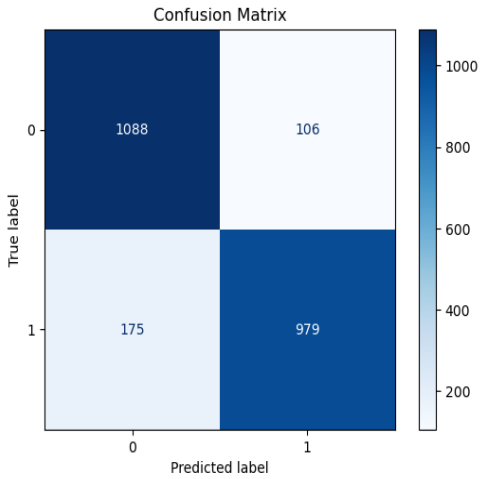

Figure 6: Confusion matrix of MLP

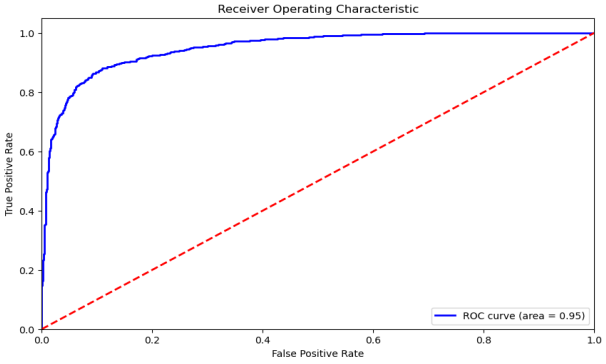

Figure 7: ROC curve of MLP

7 Data augmentation regularization

The next regularization technique implemented is data augmentation: given our filtered dataset, we perform different techniques of data augmentation (mixup augmentation, knn augmentation, gaussian mixture models augmentation, forest-based augmentation, rule-based augmentation, WGAN). In particular, for mixup, knn, gmm, forest based and rule based we defined a set of multipliers (0.5, 1.0, 1.5, 2.0) indicating the factor of augmentation inspecting how the scores of the MLP change (for example, multiplier 0.5 over a 100 samples dataset imply 50 new samples for a total 150-samples training dataset). Instead, for the WGAN augmentation we tried to augment the dataset of the 20% and of the 50%, inspecting the result and the different behavior of the MLP.

7.1 Mixup Augmentation

Mixup is a data-augmentation technique where new training examples are created by interpolating pairs of existing examples through convex combination (both for inputs and labels). Given two samples (x_i, y_i) and (x_j, y_j), one draws a mixing coefficient $\lambda \in [0,1]$ (often sampled from a Beta distribution $\text{Beta}(\alpha, \alpha)$) and forms a synthetic example as follows:

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \qquad \tilde{y} = \lambda y_i + (1-\lambda) y_j.$$

Among the list of multiplier named before, the best one found is 0.5 for totally 6845 new samples. For the filtration of the augmented dataset we followed the same approach applied in the EDA part: we identified 311 outliers.

Training an MLP with the same hyper-parameters of the ones found in section 6, gave a trainig score of 0.8477, a validation score of 0.8494 and a test score of 0.8517 and an ROC curve with area 0.94, decreasing significantly the scores found before

7.2 KNN augmentation

KNN Augmentation is a data-augmentation technique where synthetic examples are generated by interpolating each sample with its nearest neighbors. Given a dataset of samples (x_i, y_i), for each sample x_i:

1. Find the set of its K nearest neighbors $\{x_{j_1}, x_{j_2}, \ldots, x_{j_K}\}$ in the input space following a distance function.
2. Randomly select one neighbor (x_j, y_j) from this set.
3. Draw a mixing coefficient $\lambda \in [0,1]$
4. Form a synthetic example:

$$\tilde{x} = \lambda x_i + (1-\lambda) x_j, \qquad \tilde{y} = \lambda y_i + (1-\lambda) y_j.$$

By interpolating each sample only with one of its K nearest neighbors, KNN Augmentation encourages the model to be locally consistent in regions of high data density.

In this case, the best mutiplier was 1.0 for 13690 new samples and the after the filtration 384 outliers were removed. The MLP trained on this augmented dataset had a training score of score, validation score 0.8732 and test score 0.8773 with an ROC curve area 0.95; hence, even if the performance are similar to the non augmented dataset ones, the MLP trained on the original filtered dataset still is slightly better.

7.3 Gaussian Mixture Models

In Gaussian Mixture Model (GMM) synthetic examples are generated by fitting a Gaussian mixture to the data distribution of each class and then sampling new points. Concretely, for each class c:

1. Let $\{x_i^{(c)}\}$ be the training samples inside class c.
2. Fit a GMM with K components to this class's data:

$$p(x \mid c) \;=\; \sum_{k=1}^{K} \pi_k^{(c)} \, \mathcal{N}\big(x \mid \mu_k^{(c)}, \Sigma_k^{(c)}\big),$$

where $\pi_k^{(c)}$ are the mixing proportions, $\mu_k^{(c)}$ the means, and $\Sigma_k^{(c)}$ the covariances.

3. To generate a synthetic example $(\tilde{x}, \tilde{y} = c)$:

Draw a component index $z \sim \text{Categorical}\big(\pi_1^{(c)}, \ldots, \pi_K^{(c)}\big)$,

Draw $\tilde{x} \sim \mathcal{N}\big(\mu_z^{(c)}, \Sigma_z^{(c)}\big), \quad \tilde{y} = c.$

By sampling $\tilde{x}$ from the learned Gaussian mixture for each class, GMM Augmentation produces realistic, class-conditioned synthetic data that lies on the underlying data manifold.

For this techinique the best multiplier was 2.0 for totally 27380 new samples. After the filtration we removed 435 outliers and the MLP had training score 0.8732, test score 0.8613 and validation score 0.8603 with ROC curve with area 0.93. So, even in this case, the MLP trained on the original dataset is better.

7.4 Forest Based Synthesis

In Forest-Based Augmentation synthetic examples are generated by leveraging the structure of a trained ensemble of decision trees (e.g., a random forest). Let $\{(x_i, y_i)\}$ be the original dataset and let $\mathscr{F} = \{T_1, T_2, \ldots, T_T\}$ be a random forest trained on this data. Each tree T_t partitions the feature space into a set of disjoint leaves $\mathscr{L}^{(t)}$. For each leaf $\ell \in \mathscr{L}^{(t)}$, we can compute the empirical mean $\mu_\ell^{(t)}$ and covariance $\Sigma_\ell^{(t)}$ of the samples falling into that leaf.

To generate a synthetic example:

1. Select a tree index $t \sim \text{Uniform}(\{1, \ldots, T\})$.
2. Sample a leaf ℓ from T_t with probability proportional to $|\{x_i \mid x_i \text{ falls in leaf } \ell\}|$.
3. Draw $\tilde{x} \sim \mathcal{N}\big(\mu_\ell^{(t)}, \Sigma_\ell^{(t)}\big)$.
4. Assign $\tilde{y}$ as the majority class of samples in leaf ℓ.

Formally, let

$$p_t(\ell) \;=\; \frac{\big|\{x_i : x_i \in \ell\}\big|}{\sum_{\ell' \in \mathscr{L}^{(t)}} \big|\{x_i : x_i \in \ell'\}\big|},$$

and let $\mathcal{N}(\mu_\ell^{(t)}, \Sigma_\ell^{(t)})$ be the Gaussian approximation for leaf ℓ. Then a synthetic point is drawn by first choosing $t \sim \text{Uniform}(1, \ldots, T)$, then $\ell \sim \text{Categorical}(p_t)$, and finally

$$\tilde{x} \sim \mathcal{N}\big(\mu_\ell^{(t)}, \Sigma_\ell^{(t)}\big), \quad \tilde{y} \;=\; \text{mode}\big\{y_i : x_i \in \ell\big\}.$$

By sampling in this way, forest-based augmentation generates new examples that lie in high-density regions of the original data manifold.

The best multiplier for this data augmentation technique is 1.5, for totally 20535 new samples. After the filtration 554 outliers were removed and the MLP had training score 0.8452, validation score 0.8556 and test score 0.8709 and a ROC curve with area 0.95, hence the original MLP is still the best model so far.

7.5 Rule-based synthesis

In Rule-Based Synthesis the synthetic examples are generated by applying predefined transformation rules or heuristics to existing samples. Let (x_i, y_i) be an original sample and let $\mathscr{R} = \{r_1, r_2, \ldots, r_M\}$ be a set of transformation rules. Each rule $r \in \mathscr{R}$ is a deterministic or semi-deterministic function that modifies x_i while preserving its label (or adjusting it if necessary). Formally:

$$\tilde{x} \;=\; r(x_i), \qquad \tilde{y} \;=\; y_i, \quad r \in \mathscr{R}.$$

In practice, the procedure is:

1. Select an original sample (x_i, y_i).
2. Choose a rule r
3. Compute $\tilde{x} = r(x_i)$ and set $\tilde{y} = y_i$.
4. Add $(\tilde{x}, \tilde{y})$ to the augmented dataset.

For this particular case we implemented it with these features:

- **Rules are dynamically learned** from decision tree paths rather than being predefined

- For each class (binary 0/1), a decision tree partitions the feature space into hyperrectangles defined by:

$$\mathscr{R}_{\text{class}} = \bigcup_{\text{leaves}} \left\{ (f_j, \text{op}, \tau) \mid \text{op} \in \{\leq, >\} \right\}$$

- **Transformations combine noise injection and constraint projection**:
  1. Inject Gaussian noise: $\tilde{x}_{\text{noisy}} = x_i + \mathscr{N}(0, \sigma)$
  2. Project onto rule boundaries:

$$\tilde{x}_j = \begin{cases} \tau & \text{if } (f_j, \leq, \tau) \in \mathscr{R}_{\text{leaf}} \\ & \text{and } \tilde{x}_{\text{noisy},j} > \tau \\ \tau + \varepsilon & \text{if } (f_j, >, \tau) \in \mathscr{R}_{\text{leaf}} \\ & \text{and } \tilde{x}_{\text{noisy},j} \leq \tau \\ \tilde{x}_{\text{noisy},j} & \text{otherwise} \end{cases}$$

- **Label preservation** is guaranteed through class-specific rule application

In this case the best multiplier is 2.0 generating 27380 new samples. After the filtration 564 outliers were removed and the MLP gave training score of 0.8638, validation score of 0.8702 and test score 0.8833 with an ROC curve with area 0.95, having performances comparable to the ones of the MLP trained over the original dataset.

7.6 WGAN augmentation

As a last step, we explored data augmentation using Wasserstein Generative Adversarial Networks (WGANs). This approach was motivated by the limitations of traditional techniques such as mixup, which didn't ensure the preservation of critical physical constraints—most notably the invariant mass of the Z boson. In contrast, WGANs allowed us to generate synthetic data that more accurately respected the underlying distribution of the original features. We trained separate WGANs on the Zee and Zmumu subsets of the training set. The generated samples exhibited high fidelity to real data (i.e. they showed a really low wasserstain distace, approximately of 0.05). A crucial step after the new data generation and creation of the augmented dataset was applying another time the outlier filtering step, as the data generated by the WGANs indeed had some outliers that we removed to ensure compatibility with the previously cleaned dataset.

Experiments showed that a moderate augmentation (20% led to a small but measurable improvement in our best MLP model's performance (approximately +0.001). However, when the augmentation ratio was increased to 50%, the performance got worse, likely due to distributional drift or reduced diversity in the generated data.

This underscores the importance of carefully balancing real and synthetic data when applying generative models for augmentation in high-precision tasks like particle classification.

7.7 Best Data Augmentation Technique

In order to find the best data augmentation technique we watch the validation score of all models trained so far and we pick the best one. If there is a tie or two models have similar performances, we inspect also the testing scores.

8 Dropout regularization

In order to train a MLP implementing also the dropout, we had to change training library from sk-learn to TensorFlow as this regularization technique is not included in the sk-learn library. In particular, we set the hyper-parameters of the MLP equal to the optimal ones found before, and we added similarly a part finding the best dropout rate. We applied it both over the original dataset after filtration and on the dataset augmented using the best regularization technique found in section 7.7.

When the MLP was trained over the original dataset after the filtration, the best dropout rate found by the model is 0.2. The classifier got a training score of 0.8941, validation score of 0.8966 and a test score of 0.8944 with an ROC curve with area 0.96 increasing significantly the results for the best model.

Doing the same over the augmented dataset, the result was a training score of 0.8709, validation score of 0.8855 and test score of 0.8786 with ROC curve with area 0.95

9 Globally Best Classifier

Globally, among of all results, the best one is the MLP trained over the filtered dataset without data augmentation but using the dropout regularization as it gives validation score of 0.8982 and test score as 0.8952.

10 Conclusion

In conclusion, the WGAN architecture generate good enough synthetic dataset and the MLP classifier, under the good regularization techniques and hyper-parameters is a good enough binary classifier, in particular using dropout technique.