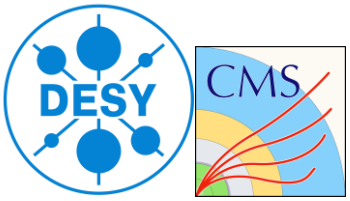


# Reconstruction of Hadronically Decaying Top Quarks with Unsupervised Self-Organizing Maps

Jawaher Altork, Marmara University, Istanbul, Turkey

Supervised by: Freya Blekman, Matthias Komm, Gabriele Milella

September, 7, 2022

**Abstract**

Machine learning (ML) has become very popular over the past few decades in practically every area. Concerning particle physics, ML has been proved as a useful resource to harvest the potential of the Large Hadron Collider (LHC) and its detectors to the fullest. The main advantage provided by ML is the reduction of time and effort put into analyzing data recorded by experiments, while simultaneously enhancing performance. The work aims to use ML, specifically unsupervised self-organizing maps, to identify resolved top decays from jets, focusing in the separation between signal and background events. Such an algorithm is foreseen to be applied in searches involving top quarks such as having resonances decaying into a top quark pair. The work started with supervised neural network as a baseline to identify the feature importance and we were able to reduce the number of features in the dataset by calculating the first order gradient of the input data with respect to the loss function. The model performance was good and promising. We were able to reject 97% of background while keeping 96% of signals. After that, unsupervised self-organizing maps (SOM) technique was applied to the data and we were able to classify jets as true top events and background events. The SOM has a descent performance but not yet matching the supervised method as expected.

# Contents

# 1. Introduction

The ability of quantum field theories to explain macroscopic and microscopic phenomena, such as quantum Hall effect and quantum electrodynamics, has been enormously successful. One of the most thoroughly examined ideas in physics is provided by the Standard Model of particle physics (SM). The SM is a quantum field theory describing the interactions of the elementary particles, providing affiliated equations of motion. Various experiments have confirmed predictions of the Standard Model at very high precision. The Standard Model, however, is unable to encompass all of physics. Gravity, or general relativity, cannot yet be taken into account in the context of a quantum field theory. Even the nature of dark matter particle state, which is strongly suggested to exists to explain several cosmological observations such as the rotation speed of galaxies, is not included in the theory. Therefore, particle colliders like the Large Hadron Collider (LHC) are built to test the current state of theory and search for new phenomena as well as new particles that are not yet included. Precision measurements of the Standard Model's parameters are essential for both. In particular, to test predictions of the theory, but also to be able to understand background processes and final states of searches. So, one of the most important tasks in LHC analyses is the reconstruction of physical objects such as electrons, muons and jets [1].

In the context of high precision SM test, as well as in searches for new particles, Top quark plays a special role in the SM. It is the heaviest particle of all known elementary particles; it has a mass about 40 times heavier than the b-quark and it is heavier than the W boson. The top quark decays to bottom quark and w boson with a branching fraction close to 100%. Furthermore, the large top quark mass implies a coupling close to 1 to the Higgs boson, thus establishing a privileged link to the Higgs sector. Also, due to its short lifetime, the top quark decays before it hadronized, passing its spin information on to its decay products. Therefore, it is possible to measure observables that depend on the top quark spin, providing a unique environment for tests of the standard model and for new physics searches.

Machine learning techniques have been used extensively in several domains of Science and Engineering for decades. These powerful tools have been applied also to the domain of high-energy physics, in the analysis of the data from particle collisions, for years already. This is the case for the CERN LHC, a broad range of topics have been addressed using machine learning, such as anomaly detection of beam position monitors, optimization of the collimation system, b-tagging and Higgs discovery.

The work entails the development of a novel unsupervised reconstruction algorithm of hadronically decaying top quarks into 3 resolved jets and comparing the results to a traditional supervised approach. Unsupervised learning applied to top reconstruction represent a novelty never used before and it has the advantage that it can be directly trained with data with less systematics.

The structure of this work is built as follows: Firstly, an overview of the Large Hadron Collider and the Compact Muon Solenoid (CMS) experiment, followed by the reconstruction of physical objects and the dataset used, respectively. Next, an introduction to machine learning and the algorithms used are presented. The development of the algorithm is shown afterwards with the results and conclusions.

## 2. Large Hadron Collider and the CMS detector

The Large Hadron Collider (LHC) [3] is a circular particle collider with a circumference of 26.7 km operating at CERN, Switzerland. At the LHC, protons, or in special runs heavy ions, are accelerated in opposite directions along the ring to perform scattering experiments at high energies. Protons were accelerated to an energy of 6.5 TeV during the second data-taking period and then brought to collision with a center-of-mass energy of $\sqrt{s}$= 13 TeV. Protons are taken from a hydrogen source and accelerated via superconducting cavities in various linear and circular accelerators before injected into the LHC ring. The protons are organized in bunches with approximately 100 billion protons per bunch. The two proton beams are collided at four interaction points inside the ring, each being the center of one experiment. Besides CMS, which is described below, another multi-purpose detector with a broad physics program operates at the LHC, called ATLAS. Additionally, there are two, more specialized, experiments: LHCb, with a focus on b-physics and ALICE, mainly aiming at research of quark-gluon plasma in heavy ion collisions.

The Compact Muon Solenoid (CMS) experiment is a multi-purpose detector at the LHC. It is designed to measure momentum and energy of stable particles produced in proton-proton interactions. The CMS detector consists of different subsystems, namely trackers, calorimeters, solenoid and muon system, aimed to reconstruct and identify different particles (see Figure 2.1). The fact that different particles leave different signatures in the detector is utilized to identify and distinguish particles. The central feature of the CMS apparatus is a superconducting solenoid of 6 m internal diameter, providing a magnetic field of 3.8 T. The silicon pixel and strip tracker, the crystal electromagnetic calorimeter (ECAL) and the brass-scintillator hadronic calorimeter (HCAL) are located within the solenoidal field volume. Muons are measured in gas ionization detectors embedded in the steel return yoke outside the solenoid. The design of CMS was chosen to cover a wide range of physics programs. Nevertheless, a focus was set on the discovery of the Higgs boson, which was announced in 2012 by ATLAS and CMS [2].

Events of interest are selected using a two-tiered trigger system. The first level of the trigger is implemented in hardware, and selects events containing detector signals consistent with an electron, photon, muon, jet, or missing transverse energy. The second level, implemented in software. A more detailed description of the CMS detector, together with a definition of the coordinate system used and the relevant kinematic variables, can be found in Ref [3].

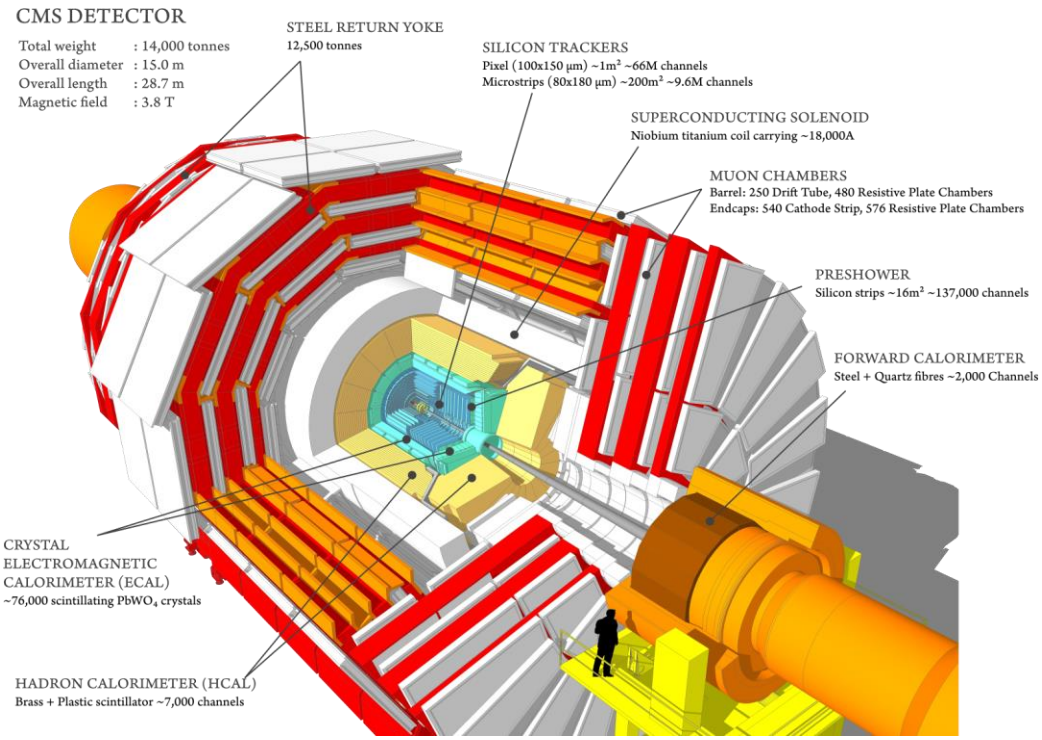


Figure 2.1: Sketch of the CMS detector and its subsystems

## 3. Reconstruction of objects (Jets)

At a particle detector like CMS, output signals have to be converted into objects with a physical meaning to perform studies with recorded data. Therefore, algorithms have been developed to identify and reconstruct particles, returning lists of electrons, muons, photons and jets. CMS uses a special algorithm, called particle flow (PF), to combine information from every detector system, following the complete path of a particle through the detector. It makes use of the various signatures different particles leave in the detector [5].

Muons are detected in the inner tracker and muon system, electrons are detected in the tracker and ECAL, charged hadrons are reconstructed using the tracker and the HCAL, and photons and neutral hadrons only leave energy in the ECAL and HCAL, respectively (see Figure 3.2). The particle flow algorithm utilizes these different signatures and combines information of all subdetectors. The goal is, to assign all showers from charged particles to their tracks, leaving only energy deposits originating from photons and neutral hadrons.

A jet can be defined as a collimated cone containing the fragmentation and hadronization products of partons (quarks or gluons) after a collision. They are objects used to reconstruct high momentum partons produced in proton-proton collisions. Because of confinement quarks and gluons cannot exist isolated but hadronized to a cascade of color neutral hadrons. To reconstruct the initial parton, a cluster procedure is needed to sum up all hadrons originating from it. Jet clustering algorithms are used to combine particle flow candidates in a well-defined way. Two important requirements for a jet algorithm are to be infrared and collinear safe. The first criterion means a jet should not change when including or excluding soft radiation. Collinear safety addresses collinear splitting of particles in a jet which should

also not change the jet. There are different jet algorithms but, in the CMS, the common way to cluster jets from the detected particles is to use iterative jet algorithms. Thus, particles are clustered step by step until an abort criterion is reached. The most common group of iterative jet algorithms are kT-like algorithms and Anti-kT [4] is considered as a standard jet finding algorithm in the CMS.

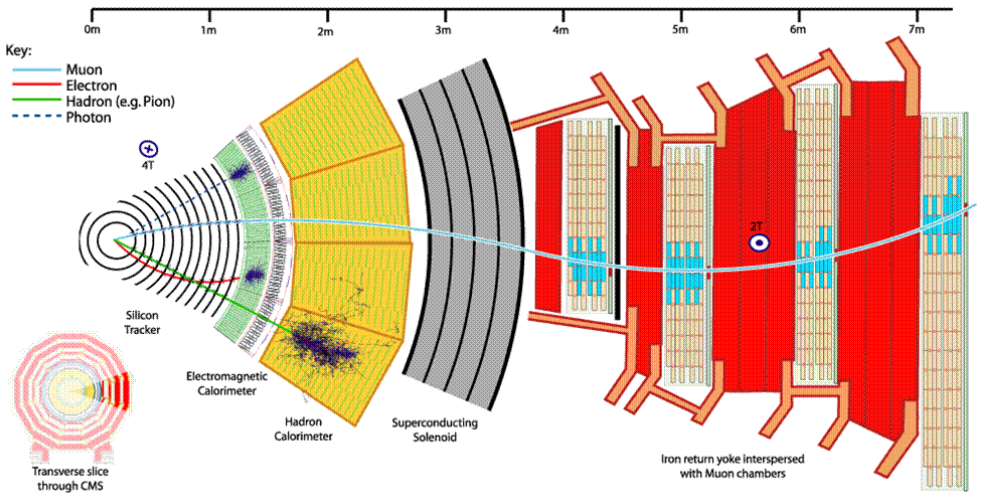


Figure 3.2: A sketch of the specific particle interactions in a transverse slice of the CMS detector, from the beam interaction region to the muon detector.

### 3.1 b-Jet Identification (b-tagging)

In analyses involving top quarks, it is useful to identify bottom quarks (b-quark) since they are in almost every case one of the top quark decay products. So, the task of identifying jets that contain b-hadrons and separating them from jets initiated by lighter quark flavors and gluons is referred to as b-tagging. The properties of the b-quark make it possible to find and tag jets originating from it. Since b-hadrons have a relatively large lifetime of about 1.5 ps, combined with time dilatation due to their velocity, they are likely to travel several millimeters in the detector before decaying. This leads to a secondary vertex at the spatial point where the b-hadron decays. The high precision tracking system of CMS makes it possible to resolve both, primary and secondary vertices. These characteristics are exploited by a b-tagging algorithm based on deep learning. Furthermore, the fraction of energy from charged hadrons in a jet and the invariant mass of all particles assigned to a secondary vertex are taken into account to distinguish them from non-b-jets [6].

## 4 Data and simulation

Monte Carlo (MC) Simulation is a mathematical technique, which is used to estimate the possible outcomes of an uncertain event. MC simulations are extensively used for various purposes in the particle physics experiments. The analysis of the LHC data at the CMS experiment requires the production of a large number of simulated events. For instance, CMS produced over 12 billion simulated events during the first data-taking period of the LHC, arranged in roughly sixty separate campaigns, each simulating certain detector characteristics and LHC operational parameters. For the purpose of this analysis, MC samples containing top events and background events are generated. The process is

generated using an event generator (e.g. pythia, powheg, madgraph); then the parton shower and hadronization is simulated with pythia8; afterwards the detector response is simulated using geant4. QCD (Quantum Chromo Dynamics), Drell Yan, and W+jets events as backgrounds in the dataset (see Figures 4.1 and 4.2). The production processes of these evets are considered constitute relevant backgrounds to searches for physics beyond the standard model.

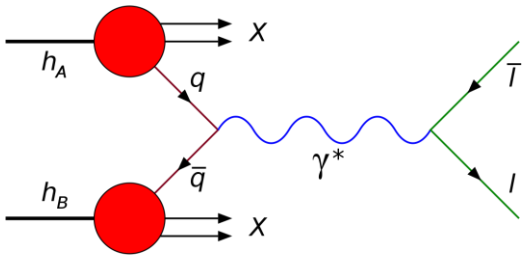


Figure 4.1: Feynman diagram of a Drell Yan process

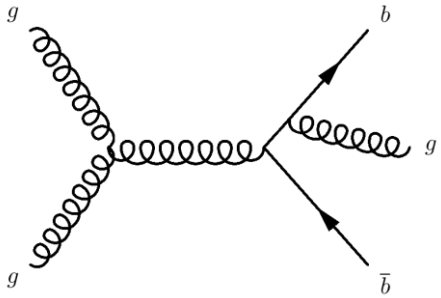


Figure 4.2: Feynman diagram of a QCD process

## 5. Machine Learning

Machine Learning (ML) is defined as the study of computer programs that leverage algorithms and statistical models to learn through inference and patterns without being explicitly programed. There are many libraries for machine learning. In this work, we used: Scikit-Learn, TensorFlow, and Keras, which are python-based libraries.

In ML, there are two kinds of data, labeled data and unlabeled data. Labeled data has both the input and the desired target values in a completely machine-readable pattern, while when a target cannot be deduced e.g. simulation-truth information, the data is unlabeled. According to this, ML can be divided into different types: Supervised and unsupervised learning.

Neural networks, also known as artificial neural networks (ANNs) are a subset of machine learning. Their name and structure are inspired by the human brain, mimicking the way that biological neurons signal to one another. The learning algorithm of a neural network can either be supervised or unsupervised.

A typical neural network is composed of neurons called units arranged in a series of layers, each of which connects to the layers on either side. They are designed to receive various forms of information that the network will attempt to learn about, recognize, or otherwise process. Other units sit on the

opposite side of the network and signal how it responds to the information it's learned; those are known as output units. In between the input units and output units more layers of hidden units are added, which, together, form the majority of the artificial brain (see Figure 5.1). Most neural networks are fully connected, which means each hidden unit and each output unit is connected to every unit in the layers either side. The connections between one unit and another are represented by a number called a weight. The higher the weight, the more influence one unit has on another [5].

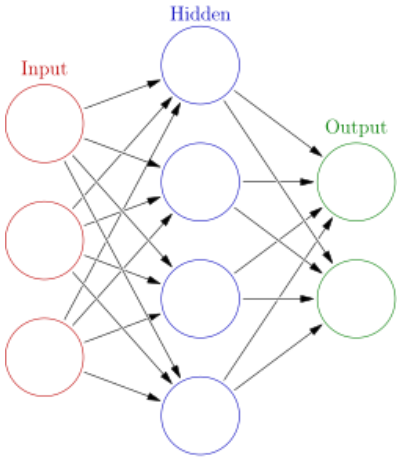


Figure 5.1: Simple architecture of neural networks

A variety of neural network structures have been developed for signal processing, pattern recognition, and so on. In this work, the neural network structures used include fully connected neural network and self-organizing maps.

**5.1 Self-Organizing Maps (SOM)**

Kohonen Self Organizing Feature Maps, or SOMs were invented by a Teuvo Kohonen, a professor of the Academy of Finland, and they provide a way of representing multidimensional data in much lower dimensional spaces - usually one or two dimensions. This process, of reducing the dimensionality of vectors, is essentially a data compression technique known as vector quantization. In addition, the Kohonen technique creates a network that stores information in such a way that any topological relationships within the training set are maintained as good as possible.

The network is created from a 2D lattice of 'nodes', each of which is fully connected to the input layer. Figure 5.1.1 shows a very small Kohonen network of 4 X 4 nodes connected to the input layer (shown in green) representing a two-dimensional vector.

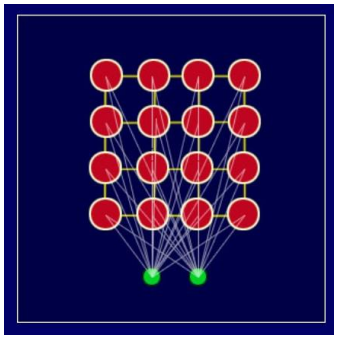

Figure 5.1.1: A simple Kohenen network

Each node has a specific topological position (an x, y coordinate in the lattice) and contains a vector of weights of the same dimension as the input vectors. That is to say, if the training data consists of vectors, V, of n samples:

$$V_1, V_2, V_3, \dots V_n$$

Then each node will contain a corresponding weight vector W, of n dimensions:

$$W_1, W_2, W_3, \dots W_n$$

A SOM does not need a target output to be specified unlike many other types of networks. Instead, where the node weights match the input vector, that area of the lattice is selectively optimized to more closely resemble the data for the class the input vector is a member of. From an initial distribution of random weights, and over many iterations, the SOM eventually settles into a map of stable zones. Each zone is effectively a feature classifier. This local classifier reflects the averaged input in its vicinity.

Training occurs in several steps and over many iterations:

1. Assign weights with the same dimensionality as the input data to each node in the 2d map.

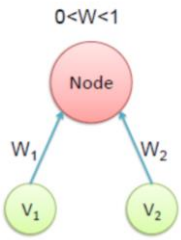


Figure 5.1.2: Initialization of the node's weights

2. Choose an input vector randomly from the set of training data. This is the competitive stage since all nodes compete against each other.
3. Iterate through all the nodes and calculate the activation distance (eg. Euclidean distance) between each node's weight vector and the current input vector. The node with the smallest distance wins and is known as the Best Matching Unit (BMU).
4. The size of the neighborhood (eg. Gaussian) of the BMU is now calculated. Any nodes found within this size are deemed to be inside the BMU's neighborhood.
5. The BMU and each of its neighboring node's weights are adjusted to make them more like the input vector. The closer a node is to the BMU, the more its weights get altered. This is the collaborative stage since the nodes in the neighborhood are collaborating with each other.
6. Repeat from step 2 for N iterations until convergence.

Looking at each step in more details,

First, to determine the best matching unit, one method is to iterate through all the nodes and calculate the Euclidean distance between each node's weight vector and the current input vector. The node with a weight vector closest to the input vector is tagged as the BMU. The Euclidean distance is given as:

$$Dist = \sqrt{\sum_{i=0}^{i=n} \sum_{j=0}^{j=m} (V_i - W_j)^2}$$

where V is the current input vector and W is the node's weight vector.

After the BMU has been determined, the next step is to calculate which of the other nodes are within the BMU's neighborhood. All these nodes will have their weight vectors altered in the next step. One can calculate what the radius of the neighborhood should be and then it is a simple application to determine if each node is within the radial distance or not.

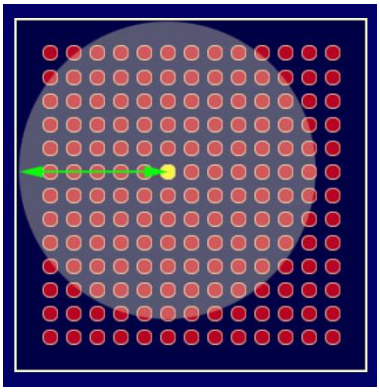

Figure 5.1.3: The BMU's neighborhood

It can be seen that the neighborhood shown in Figure 5.1.3 is centered around the BMU (colored yellow) and encompasses most of the other nodes. The green arrow shows the radius.

A unique feature of the Kohonen learning algorithm is that the area of the neighborhood shrinks over time. This is accomplished by making the radius of the neighborhood shrink over time. To do this, the exponential decay function can be used:

$$\sigma(t) = \sigma_o \exp\left(-\frac{1}{\lambda}\right) \qquad , t = 1,2,3 \ldots$$

Where sigma, σ, denotes the width of the lattice at time t and the lambda, λ, denotes a time constant. t is the current time-step (iteration of the loop). Over time the neighborhood will shrink to the size of just one node, the BMU. This shrinking decreases the collaboration between the nodes.

Now that the radius is known, it's a simple matter to iterate through all the nodes in the lattice to determine if they lay within the radius or not. If a node is found to be within the neighborhood, then its weight vector is adjusted as follows.

Every node within the BMU's neighborhood (including the BMU) has its weight vector adjusted according to the following equation:

$$W(t+1) = W(t) + \theta(t)\, L(t)\big(V(t) - W(t)\big)$$

Where t represents the time-step and L is a small variable called the learning rate, which decreases with time.

Theta, $\theta$ , represent the amount of influence a node's distance from the BMU has on its learning and is given by:

$$\theta(t) = \exp\left(-\frac{Dist^2}{2\sigma^2(t)}\right)$$

The decay of the learning rate is calculated in each iteration using the following equation:

$$L(t) = L_o sxp\left(-\frac{1}{\lambda}\right) \qquad , t = 1,2,3 \ldots$$

Finally, from a random distribution of weights and through much iteration, SOM is able to arrive at a map of stable zones. At the end, interpretation of data is to be done by human but SOM is a great technique to present the invisible patterns in the data [7][8].

## 6. Results

For the dataset, as it was mentioned before, Monte-Carlo (MC) samples are used with top in production events and background events looking into different Jet objects such as: charged Electromagnetic Energy Fraction, b jet transverse momentum, triplet jets (bjj) mass, two jets (jj) mass, Number of particles in the jet, Charge Hadron energy Fraction, etc. In total there are 53 jets objects. The total size of the input data is about 80,000 events, 50% of them are signal jets and 50% are background jets. Then, scikit-learn (train_test_split) is used to split the dataset into 70% training and 30% testing sets.

### 6.1 Supervised Neural Network: Classification

We are only interested in encoding the features that are relevant for distinguishing between top and no top decays; hence it is important to first reduce the number of features since unsupervised algorithm have no notion of the importance and will try to encode all of them (this is also true for e.g. Autoencoders).

Binary Classification with TensorFlow (Keras) model is used, which classify the events into signal jets and background jets according to their gen matching $\Delta R$ with respect to the original parton (Signal jet has $\Delta R$ < 0.4 while background jet has $\Delta R$ > 1.5) as shown in Figure 6.1.1.

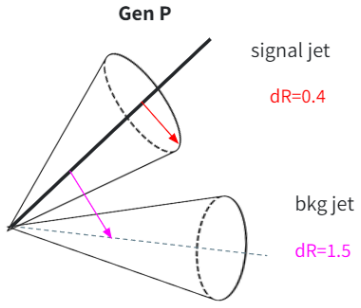


Figure 6.1.1: The two jets are classified into signal and background (bkg) jets according to their dR which is the radius of the jet cone.

The process of training begins with creating the fully connected neural network model with 5 hidden layers [100,90,70,50,50]. SELU (Scaled Exponential Linear Units) is used as an activation function, and a 10% dropout layer is used as well (as shown in Figure 6.1.2). The Dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting. After that, the model is compiled using Adam optimizer with 0.001 learning rate, and the loss function used is binary cross entropy from logits. Next is Fitting the model to the input training instances with a batch size of 256, and 100 epochs. The training was stopped at 100 epochs to prevent overfitting. Finally, predictions can be performed on the testing instances, based on the learned parameters during the fit and the reconstructed output can be evaluated using the loss function and other evaluation functions.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 28)]              0

 batch_normalization_1 (Batc  (None, 28)               112
 hNormalization)

 dense (Dense)               (None, 100)               2900

 dropout (Dropout)           (None, 100)               0

 dense_1 (Dense)             (None, 90)                9090

 dropout_1 (Dropout)         (None, 90)                0

 dense_2 (Dense)             (None, 70)                6370

 dropout_2 (Dropout)         (None, 70)                0

 dense_3 (Dense)             (None, 50)                3550

 dropout_3 (Dropout)         (None, 50)                0

 dense_4 (Dense)             (None, 50)                2550

 dropout_4 (Dropout)         (None, 50)                0

 dense_5 (Dense)             (None, 1)                 51

=================================================================
Total params: 24,623
Trainable params: 24,567
Non-trainable params: 56
_________________________________________________________________
```

Figure 6.1.2: Binary classification Keras model

In order to reduce the number of features or jets variables, the first order gradient of the input data is calculated with respect to the loss function to see how sensitive the predictions are to certain jets variables and it is shown in Figure 6.1.3.

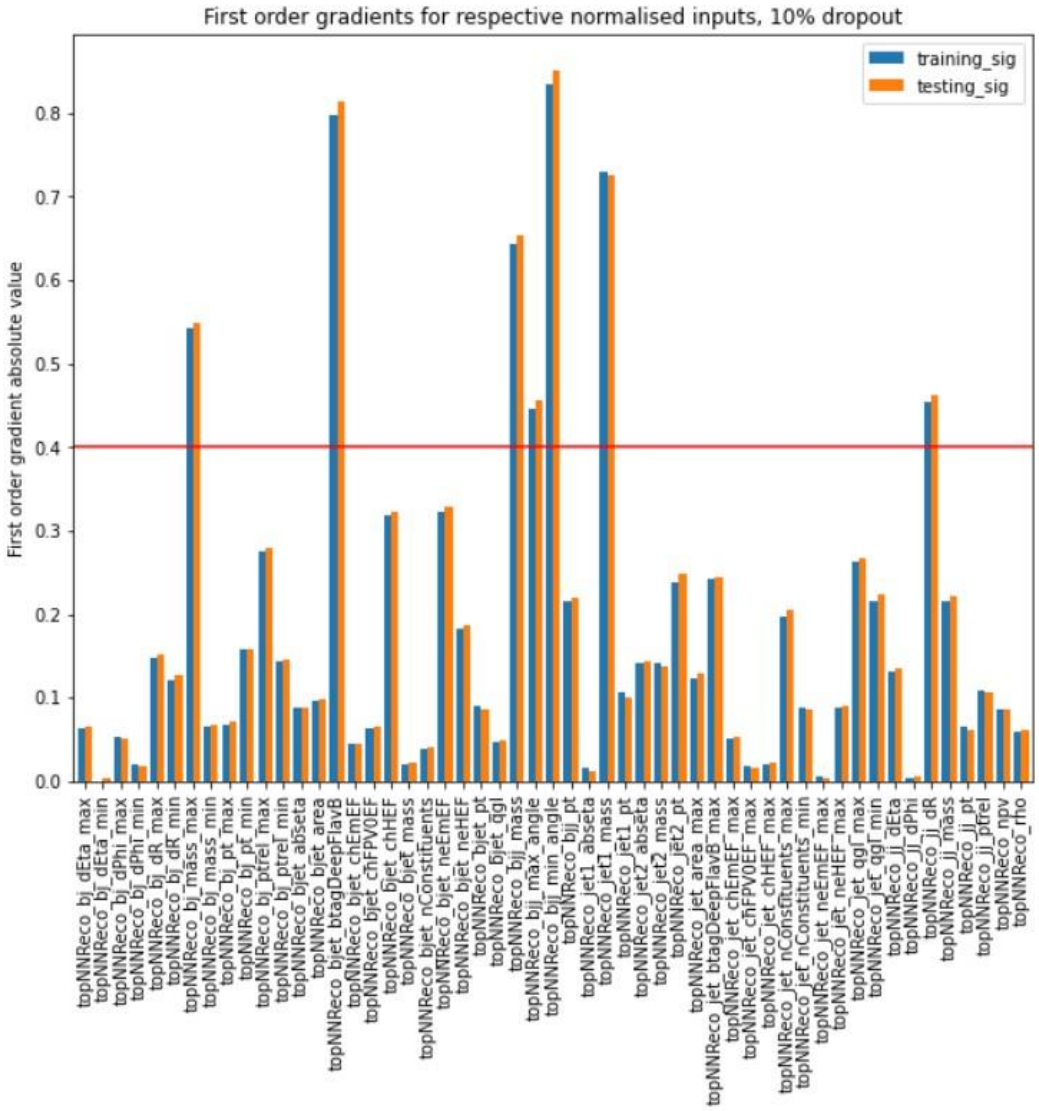


Figure 6.1.3: First order gradient for different jet variables

Then, the model is trained with the first 7 variables having the highest first order gradient, which are: bj_mass_max, bjet_btagDeepFlavB, bjj_mass, bjj_min_angle, bjj_max_angle, jet1_mass, jj_dR. After that, more variables are added, then a comparison is made by looking into different metrics e.g. train and test accuracy, train and test loss, and ROC AUC (Area Under the Curve) scores as seen in table 6.1.1. The new training data is taken with 28 variables. Since adding further variables increased the accuracy by only by 1% while not including them nearly halved the number of variables which is considered a good tradeoff.

| Evaluation | All 53 variables | 7 variables | 14 variables | 19 variables | 23 variables | 28 variables |
|---|---|---|---|---|---|---|
| Train Accuracy | 0.9012 | 0.838 | 0.870 | 0.875 | 0.882 | 0.899 |
| Test Accuracy | 0.886 | 0.834 | 0.873 | 0.873 | 0.875 | 0.885 |
| Train Loss | 0.242 | 0.367 | 0.304 | 0.292 | 0.283 | 0.247 |
| Test Loss | 0.278 | 0.373 | 0.302 | 0.304 | 0.295 | 0.273 |
| Train AUC score | 0.964 | 0.913 | 0.941 | 0.946 | 0.949 | 0.965 |
| Test AUC score | 0.952 | 0.909 | 0.942 | 0.942 | 0.949 | 0.954 |

Table 6.1.1: shows different evaluation functions for different jet variables

To evaluate the model, different metrics plots are used such as: accuracy, loss, and ROC curve. As shown in the Figures 6.1.4, 6.1.5, and 6.1.6 below.

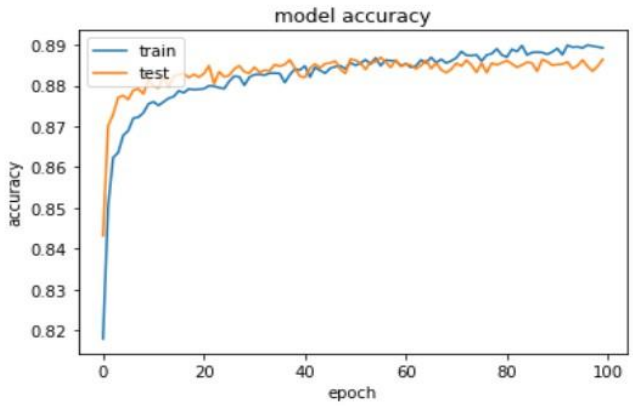


Figure 6.1.4: Accuracy graph of the proposed model

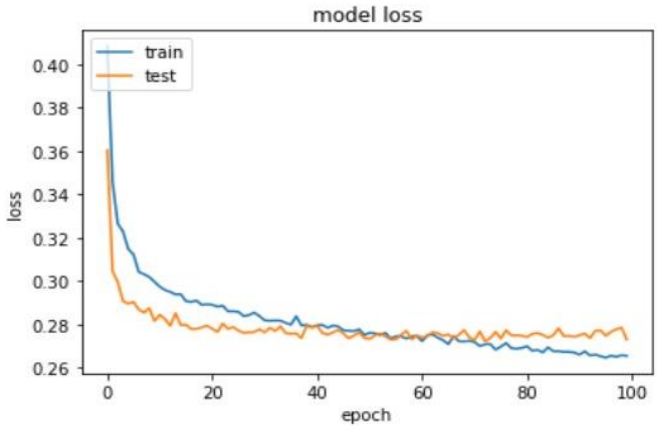


Figure 6.1.5: Loss graph of the proposed model

Training and testing sets are statistically independent and only the training dataset is used to update the NN wights. The loss is calculated on training and testing sets and its interpretation is how well the model is doing for these two sets. Unlike accuracy, loss is not a percentage. It is a summation of the errors made for each example in training or validation sets. Also, it can be seen from the plots that train and test curves are not similar and this due to dropout regulation.

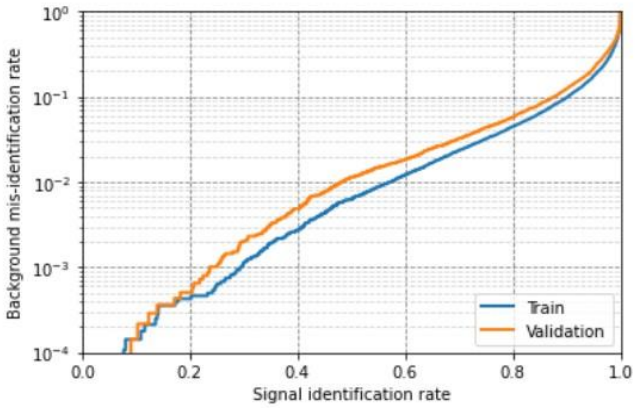


Figure 6.1.6: ROC curve for train and testing data

The model performance was good with accuracy > 0.88 and ROC AUC score > 0.96 for both training and testing sets. Now that the variables are reduced in the dataset, training with unsupervised self-organizing maps is what is next.

**6.2 Unsupervised Neural Network: Self-Organizing Maps (SOM)**

MiniSom Model is used here [9] which is a NumPy implementation of the self-organizing maps. The number of nodes is chosen to be 13 X 13 or 169 clusters. Euclidean distance is used as an activation function and Gaussian distribution is considered as a neighborhood function with a neighborhood radius of 0.3. The topology of the map is taken as rectangular.

After training, SOM produced a 2-dimensional representation of the data distributed into 169 clusters. The idea is to look into different clusters in the jj mass and bjj mass plane since it is known that if the jet comes from the top quark these two variables should correspond to the mass of w-boson and mass of the top, respectively which is shown in Figure 6.2.1.

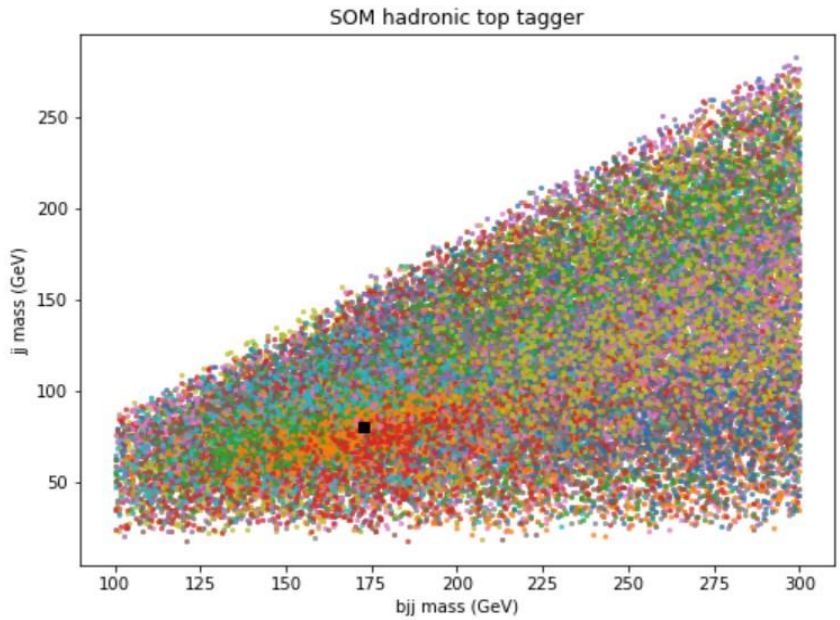


Figure 6.2.1: 169 clusters distributed in the bjj mass X jj mass plane. The black point represents the mass of the top and mass of w, respectively.

Looking at each cluster individually, in some clusters the points are distributed far away from the mass of the top so they are most likely to be considered as background-like clusters (see Figure 6.2.2) unlike other clusters where the points are clustered near the mass of the top (see Figure 6.2.3)

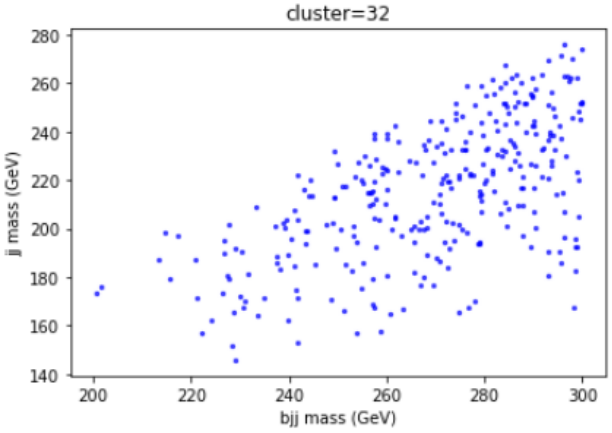


Figure 6.2.2: cluster 32 in the bjj mass X jj mass plane

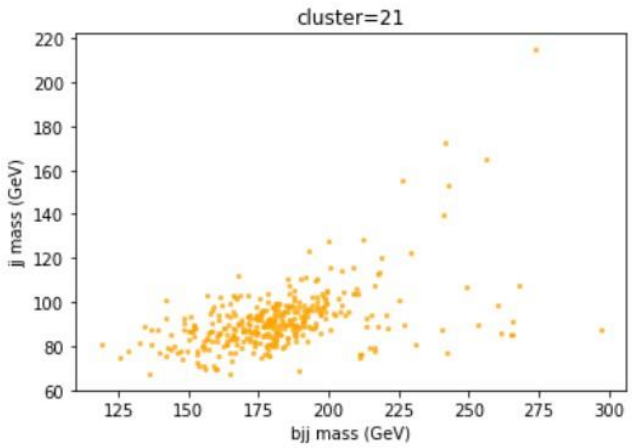


Figure 6.2.3: cluster 21 in the bjj mass X jj mass plane

In order to evaluate how good each cluster is to identify true top events; the radii R is calculated for each cluster which is defined as the distance between each point in each cluster from the top mass and w mass:

$$R = \sqrt{\left|(x - top\_mass)\right|^2 + \left|(y - w\_mass)\right|^2}$$

Where;

x = bjj mass, y = jj mass, top_mass = 172.5, w_mass = 80.4.

The Figure below shows a histogram of different R values for different clusters.

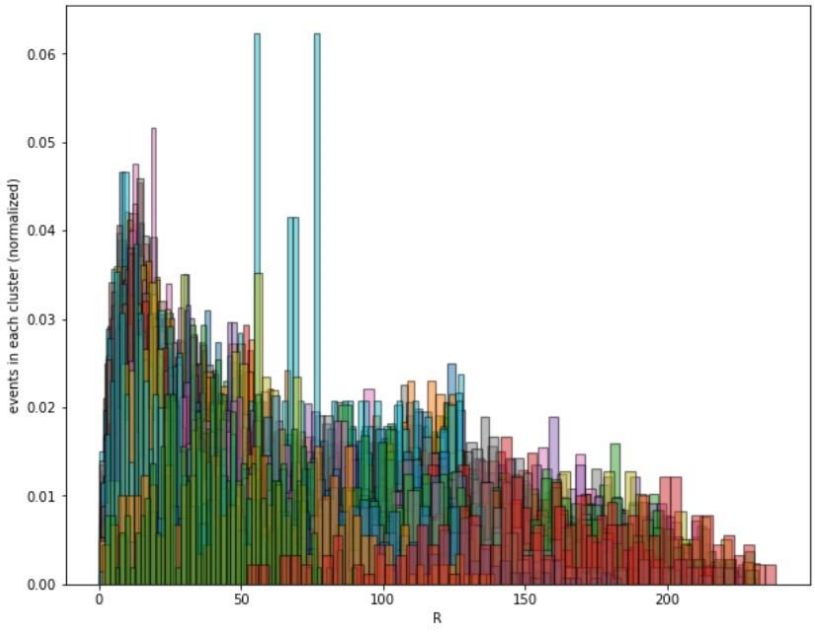


Figure 6.24: Histogram of the R values for each cluster

The clusters having R values close to zero can be considered as signal-like clusters.

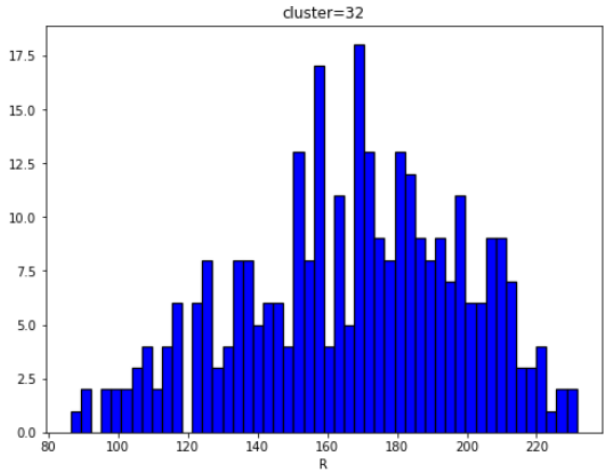


Figure 6.25: Histogram of the R values in cluster 32

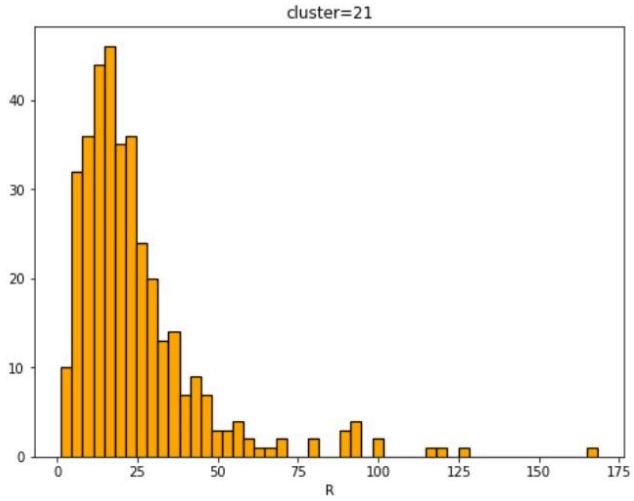


Figure 6.26: Histogram of the R values in cluster 21

After that, the mean values of R are calculated for each cluster and is plotted with the signal efficiency which is defined as:

$$Signal\ efficiency\ = \frac{actual\ top\ events\ in\ each\ cluster}{number\ of\ events\ in\ each\ cluster}$$

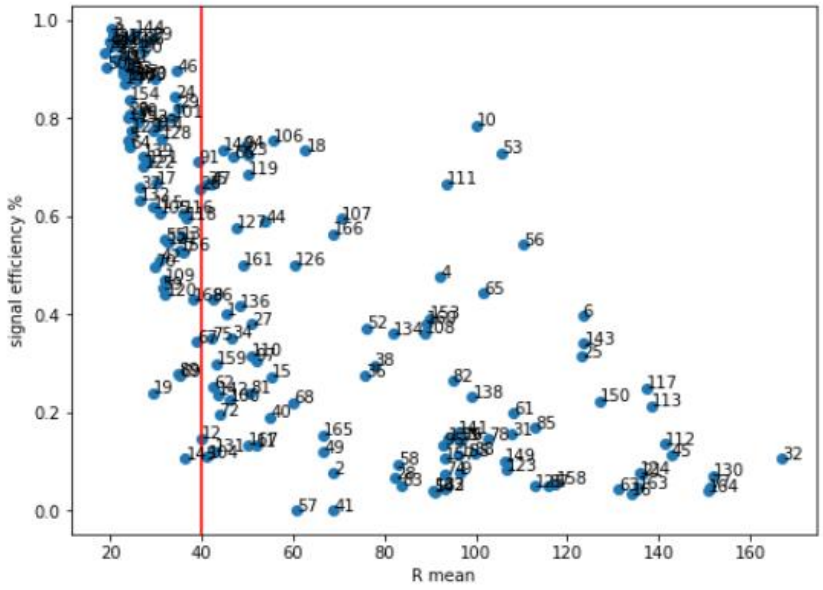


Figure 6.27: R mean vs signal efficiency for each cluster (the numbers written above each point represent the cluster number from 0-168)

The idea then is to perform a cut on the mean value of R and classify all the clusters into signal and background clusters. In Figure 6.27, the threshold for R mean is chosen to be 40. So, every cluster that has R mean value less than 40 is classified as signal-like cluster. For this threshold, 50% of the training data is distributed in these clusters with 86% of the data having signal efficiency above 0.5.

Next is comparing the classification approach with the self-organizing maps. One way to do it is to sum up all the clusters having R less than 40 into one single cluster then calculate the signal and background efficiencies for the resulting cluster. The resulting point can be placed in the classifier ROC curve and is shown as a red point in Figure 6.2.8.

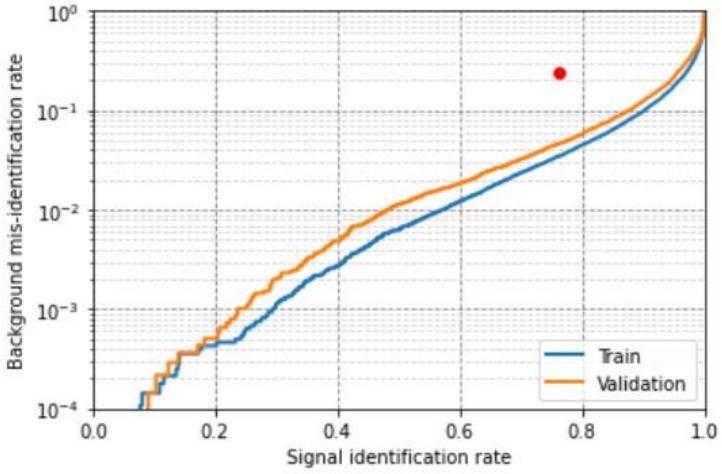


Figure 6.2.8: ROC curve of the classifier with point resulting from the SOM and the coordinate of the point is (0.762,0.264) = (signal selected/total signal), (background selected/total background)

## 7. Summary and Conclusion

Identification of resolved top decays from three jets combination using neural networks has been presented. The work was performed using Monte Carlo samples containing events with and without top in production. Classification algorithm is used to reduce the number of features. The classifier performance was good with 0.88 accuracy and Roc AUC > 0.96. Then, the dataset is trained with unsupervised self-organizing maps where the data is mapped into different clusters. By calculating the mean of the radii for each cluster, they can be classified into signal-like or background-like clusters. The SOM had a descent performance but not yet matching the supervised method as expected. As a next step for this study is to find a better way to compare the supervised and unsupervised approaches, and then finally implement the method to data.

## 8. References

[1] CMS Collaboration, “Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC”, arXiv:1207.7235.

[2] ATLAS Collaboration, “Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC”, arXiv:1207.7214

[3] CMS Collaboration,“LHC machine technical design reports (TDR)”, JINST 3 S08001, JINST 3 S08002, JINST 3 S08003, JINST 3 S08004, JINST 3 S08004.

[4] “The anti-k_t jet clustering algorithm”, Matteo Cacciari, Gavin P. Salam, Gregory Soyez, arXiv:0802.1189

[5] CMS Collaboration, “Particle-flow reconstruction and global event description with the CMS detector”, JINST 12 (2017) P10003.

[6] “Identification of b-quark jets with the CMS experiment”, Journal of Instrumentation 8 (2013), no. 04, P04013.

[7] “Machine learning with neural networks”, Bernhard Mehlig, arXiv:1901.05639v4 [cs.LG] 27 Oct 2021.

[8] “The Self Organizing Maps”, Teuvo Kohonen, https://sci2s.ugr.es/keel/pdf/algorithm/articulo/1990-Kohonen-

[9] “JustGlowing/minisom”, GitHub, https://github.com/JustGlowing/minisom.git