$t\bar{t}H$ Events Classification with Graph Neural Networks in $2L(SS) + 1\tau_{had}$ Channel

Paramott Bunnjaweht[1] and Nello Bruscino[2]

[1]Chulalongkorn University
[2]Sapienza Università & INFN Roma

September 5, 2023

1 Introduction

Some of the problems of particle physics demand an optimal distinction between the null hypothesis (described by a "signal" sample) and the alternative one ("background" sample). Algorithms have been created to automatically perform the task by directly analyzing high-level information such as the 4-momentum of all the objects produced, for instance, by proton-proton collisions at the Large Hadron Collider (LHC), and properties of specific objects like the b-tagging score of the hadron jets. Among these are machine learning algorithms that are first "trained" on data under two or more known hypotheses (aka, supervised mode). Examples are Decision Trees (with subsequent improved counterparts such as Boosted Decision Trees (BDTs)) and Artificial Neural Networks (ANNs).

In this project, we aim to test new machine learning algorithms (Graph Neural Networks) to improve the performances of the current $t\bar{t}H$ analysis carried by the ATLAS Collaboration at a center-of-mass energy of 13 TeV. We focus on the final state (channel) with two lepton same-sign plus one hadronic tau lepton. Precisely, the processes that we study are:

- The production of a Higgs Boson with a top quark pair ($t\bar{t}H$) as signal
- The production of a W Boson with a top quark pair ($t\bar{t}W$) as background
- The production of a Z Boson with a top quark pair ($t\bar{t}Z$) as background

We attempted to design a Graph Neural Networks (GNNs) model that exploits the relational structure of variables in form of a graph to distinguish signal events from background events with performance possibly exceeding that of BDTs. In this project, we trained and compared the performance of BDTs, Fully Connected ANNs and GNNs.

2 Variables and Graph Structure

A collision event and its subsequent analysis produce data about the properties of the objects. In the events we studied, there are 14 objects considered, 10 of which are hadron jets [1], 2 are light leptons, 1 for a tau lepton and 1 being "Missing Energy in the transverse plane (MET)", indicating the portion of energy missing from the observation, supposedly carried by undetected particles like neutrinos. All objects apart from MET have at least 4 variables specifying their 4 momentum (E, p_T, ϕ, η), namely energy, transverse momentum, polar angle and pseudorapidity (amounting to 52 variables). In addition:

- Every jet provides a variable for its b-tagging [2] score;
- Every light lepton contains information on its charge and type in a single variable;
- The tau lepton contains a variable specifying its charge;
- MET contains 5 variables: The energy missing in the transverse plane of the event and its transverse plane angle, the invariant mass of leptons and the missing object, the invariant mass in the transverse plane of leptons and the missing object, and the invariant mass in the transverse plane of the first lepton and the missing object.

There are also variables that describe the relationship between each object;

- 45 variables are the angular distance between 2 jets, defined by $\Delta R = \sqrt{(\Delta\phi)^2 + (\Delta\eta)^2}$
- 20 variables are the distance between a jet and a light lepton
- 10 variables are the distance between a jet and the tau lepton
- 2 variables are the distance between a light lepton and the tau lepton
- MET has one variable in relation to the jet with the highest transverse momentum, being the difference of transverse angle.
- For the light leptons pair, these are the distance between the two leptons, the sum of their invariant mass and the transverse momentum of two leptons.

These variables form a graph consisting of objects or "nodes" linked together by "edges", with object variables prescribed to each node in the form of a vector called node attributes and relational variables acting as edge attributes.

On top of these, there are variables that do not belong to any object or form a connection between two objects, called "global variables". This includes variables such as the number of jets produced, the maximum pseudorapidity, and so on.

In total, there are $60 + 81 + 18 = 159$ variables.

[1] There are events with less than 10 jets, the variables of the "placeholder" jets are set to zero. There are also events with more than 10 jets, the variables of such jets will not be used in classification

[2] The b-tagging score is the output of a multivariate classifier trained to distinguish jets originated from b-quarks w.r.t. jets from c- or light-quarks

3 Learning Algorithms

3.1 Boosted Decision Trees

Decision Trees are a popular machine learning algorithm that operates by making decisions based on asking a series of questions. By evaluating the features of data, they recursively split the dataset into subsets, aiming to achieve subsets (or leaf nodes) where all data points belong to a single class or have a similar value. Boosted Decision Trees enhance the basic decision tree algorithm by building an ensemble of trees in a sequential manner. In BDTs, each data point in the dataset is assigned a weight, and after each tree is built, these weights are adjusted to emphasize the points that were mis-classified by the previous tree. This ensures that subsequent trees in the sequence focus more on the harder-to-classify examples. The final prediction is a weighted combination of the predictions from all individual trees, leading to a more accurate and robust model compared to a single decision tree. Decision trees are intuitive, interpretable, can capture non-linear relationships, and typically require less data to achieve a good performance compared to training an ANNs model.

3.2 Artificial Neural Networks

An Artificial Neural Networks (ANNs) model is, in essence, a composite function consisting of a succession of mathematical operations or layers applied to data points, with the final output being interpreted according to the intended usage. In the context of binary classification, an ANNs model transforms the input data into a number to be interpreted as the probability, or the model's confidence, that the object (in our case the event) containing this particular data point belongs to a class of objects (signal events vs background events). The simplest ANNs models are a stack of "Fully Connected" layers; matrix multiplication acting on an input data vector and non-linear "activation" functions applied element-wise to the resulting vector. ANNs models are normally trained using the Gradient Descent method, which allows all parameters in a model (for example, matrix elements in layers) to be tuned at once, improving the model.

3.3 Graph Neural Networks

Graph Neural Networks (GNNs) models are a class of ANNs where the input data is in the form of graphs. A graph with N nodes, each node having a d-length feature vector, can be mathematically represented by an adjacency matrix $A = [a_{ij}]_{N\times N}$, an N x N matrix which characterizes the connections between node i and node j (equals 1 if connected, 0 otherwise), and a "feature matrix" $F = [f_{ij}]_{Nxd}$ which contains data for each node. A layer that processes the input graph takes into account both the adjacency matrix and the feature matrix. To retrieve the signal probability of an event, The graph and its information have to be reduced in an operation called a pooling layer that aggregates the data present together, into either a smaller graph or down to a single vector, which can be processed further by Fully Connected layers to retrieve the final output. The actual form of the graph designed

is heterogeneous, meaning there are different types of nodes and edges, allowing for a more flexible arrangement of variables in a graph but the computation is more complex.

4 Methods

We trained a BDTs model, a vanilla ANNs model (consisting only of Fully Connected layers) and a GNNs model using the same set of variables to classify the events. The samples used are data generated by Monte Carlo simulations from various frameworks.

Events	Number of MC events
$t\bar{t}H$	24093
$t\bar{t}W$	6947
$t\bar{t}Z$	11437

Table 1: The number of Monte Carlo (MC) events in each event type.

We used mva-trainer, a machine learning model training framework. We can create a model by making a configuration file containing the variables used as the model's inputs, the samples used, and the model's architecture. In the case of Graph Neural Networks, the graph structure of the variables involved must be specified. This includes specifying nodes, feature vectors belonging to each node, edges connecting pairs of nodes, and edges feature vectors.

The architecture of the GNNs model is described as follows: An input graph is passed through a layer of GATConv (Graph Attention Convolution) which processes both the node attributes and edge attributes together and outputs a graph with the same structure but with transformed node attributes with a length of 10 each. Afterwards, the graph is passed into a pooling layer where all nodes' attributes are used to compute a vector which is the average value of them, reducing the graph into a single vector. Since the graph is heterogeneous, 4 different types of nodes (Jets, Leptons, Tau, MET) are passed into separate GATConv layers and are pooled separately and so there are 4 vectors of length 10 in total. Global attributes form a vector that is transformed by a single Fully Connected layer from 18 input variables to 10 output variables. The resulting vectors are directly concatenated, forming a vector of length 50 which is then passed into 2 Fully Connected layers, resulting in a single number describing the probability that an input event is a signal event.

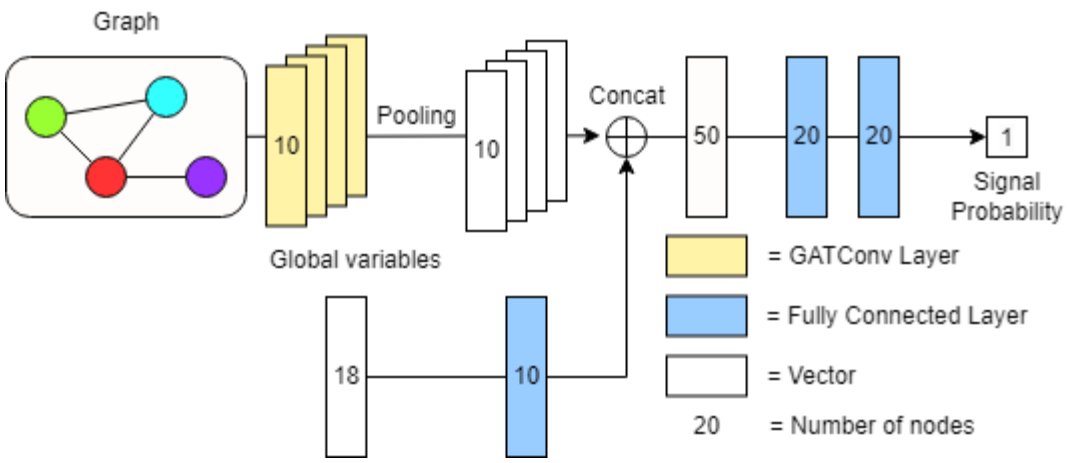


Figure 1 : The GNNs architecture used

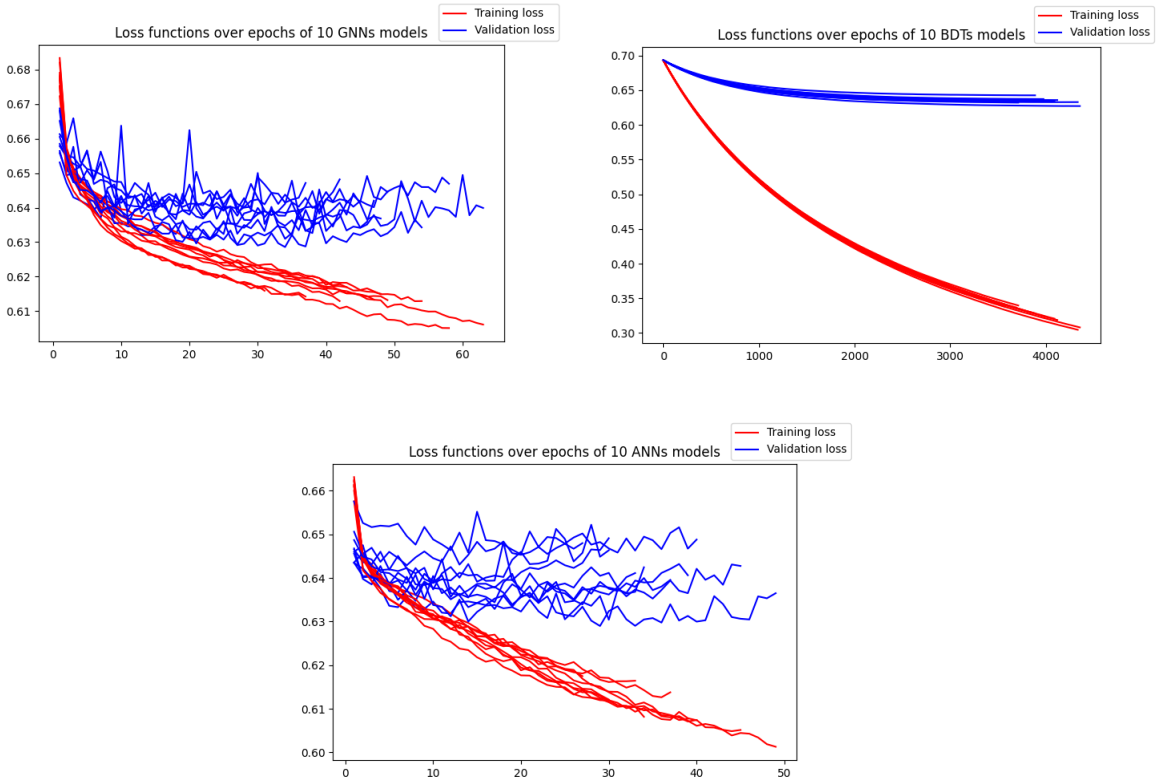
The model uses Rectified Linear Unit (ReLU) function as an activation function for every single layer except in the final layer where the sigmoid function is used instead. In training, the model is optimized with the Nadam algorithm using Binary Cross-entropy as the loss function with a learning rate of 0.002. In each training step, 200 events are processed in a batch before the model's parameters are optimized. 20 percent of the available events are set aside as validation dataset excluded from being used to train the model.

For the BDTs model, the model contains from 3000 to 4000 trees (depending on when the training performed early stopping), with the maximum tree depth of 10 and is trained using Binary Cross-entropy loss function with a learning rate of 0.001.

For the ANNs model, the model is made of 3 layers of 20 nodes Fully Connected layers with ReLU function as an activation function and a final 1 node Fully Connected layer with Sigmoid function as the activation function. The model is trained using Binary Cross-entropy loss function with a learning rate of 0.002.

5 Performance Evaluation

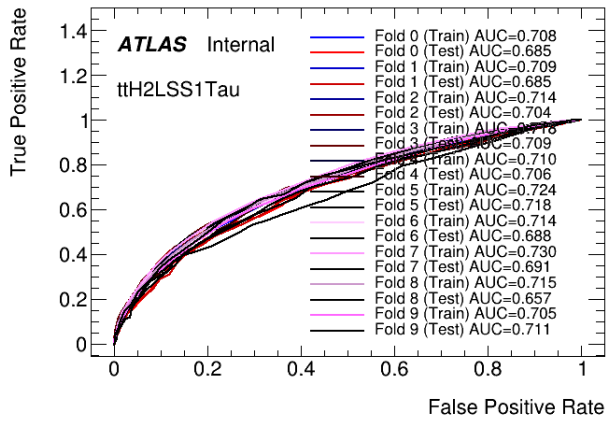
Models are trained and evaluated 10 times with the available train/test dataset divided by the k-Fold cross-validation technique.

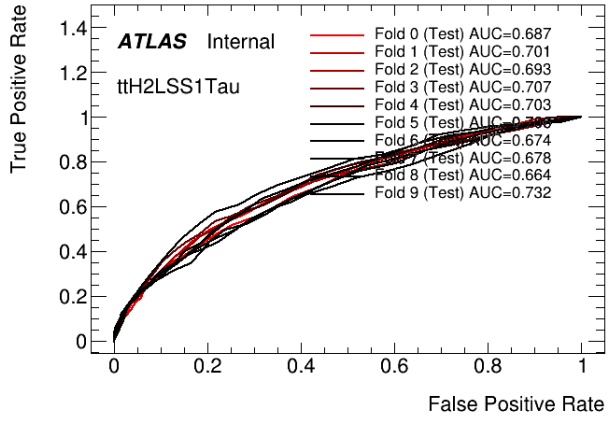


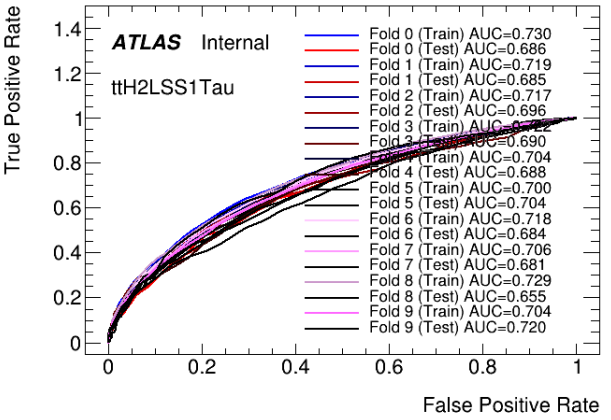
Figures 2-4 : Graph showing the loss function of GNNs, BDTs and vanilla ANNs models, respectively

It appears that overfitting becomes apparent early on in the training. The validation loss functions of all models across three model types are roughly at the same level.

To gauge a model's performance, we look at the Area Under Curve (AUC) of the Receiver Operator Characteristics curve (ROC). The ROC curve characterizes the rate of correctly predicting signal events (true positive) the model would achieve by allowing incorrect signal prediction as background events (false positive) at a rate. The higher the AUC, the better the model is.







Figures 5-7: Graph showing the ROC curves of GNNs, BDTs and vanilla ANNs models, respectively

	GNNs	BDTs	ANNs
Fold 1	0.685	0.687	0.686
Fold 2	0.685	0.701	0.685
Fold 3	0.704	0.693	0.696
Fold 4	0.709	0.707	0.690
Fold 5	0.706	0.703	0.688
Fold 6	0.718	0.706	0.704
Fold 7	0.688	0.674	0.684
Fold 8	0.691	0.678	0.681
Fold 9	0.657	0.664	0.655
Fold 10	0.711	0.732	0.720
Average	**0.695**	**0.694**	**0.689**
S.D	0.018	0.020	0.017

Table 2: The AUC of models evaluated using the test dataset.

6 Conclusion

We found that the Graph Neural Networks' performance does not significantly improve from the Boosted Decision Trees. It is unclear how we can improve a model's performance, but it is likely that increasing the number of the samples will improve GNN's performance which normally require a large amount of data to outperform BDTs and other traditional machine learning methods.

References

[1] *Analysis of $t\bar{t}H$ and $t\bar{t}W$ production in multilepton final states with the ATLAS detector*. Tech. rep. All figures including auxiliary figures are available at https://atlas.web.cern.ch/Atlas/GROU CONF-2019-045. Geneva: CERN, 2019. URL: http://cds.cern.ch/record/2693930.

[2] Gareth James et al. "Tree-Based Methods". In: *An introduction to statistical learning: With applications in Python*. Springer, 2023, pp. 331–366.

[3] Steffen Korn. *MVA-trainer*. 2023. URL: https://gitlab.cern.ch/skorn/mva-trainer/.

[4] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: 1710.10903 [stat.ML].