



Using Machine Learning Techniques for the Analysis of Four-Top-Quark Production

John Lawless, Iowa State University, United States

Supervisors: Philipp Gadow and Krisztian Peters

September 10, 2021

Abstract

In the search for heavy resonances in four-top-quark final states in pp collisions with the ATLAS detector, the precision of the mass resolution is of vital importance. One contributing factor to the measurement is the proper identification of the top quarks as either having participated in the resonance decay, labelled resonance, or having not participated, labelled spectator. In this analysis, we researched whether using machine learning techniques could help identify resonance versus spectator top quarks with better accuracy than human decided cuts. We found that in Monte Carlo data of reconstructed top quarks the four machine learning techniques outperformed human cuts in almost all cases.

17	Contents	
18	1 Introduction	3
19	2 Dataset	4
20	3 Methods	6
21	3.1 Simple Cuts	6
22	3.2 KNN	6
23	3.3 BDT	6
24	3.4 Neural Networks	7
25	3.5 Graph Neural Networks	7
26	4 Measurement	8
27	4.1 Simple Cuts	8
28	4.2 KNN	9
29	4.3 BDT	9
30	4.4 Neural Networks	9
31	4.5 Graph Neural Networks	10
32	5 Conclusions	11

1 Introduction

The current analysis "Search for heavy resonances in four-top-quark final states in pp collisions at $\sqrt{s} = 13$ TeV with the ATLAS detector" is a model independent search of the exciting final state of four top quarks. The analysis is using ATLAS Run-2 data collected in proton-proton collisions in the LHC. This final state has a high discovery potential for heavy particles that couple strongly to top quarks. Due to the search being model independent, the final analysis will look at multiple different possibilities for resonance peaks in the data.

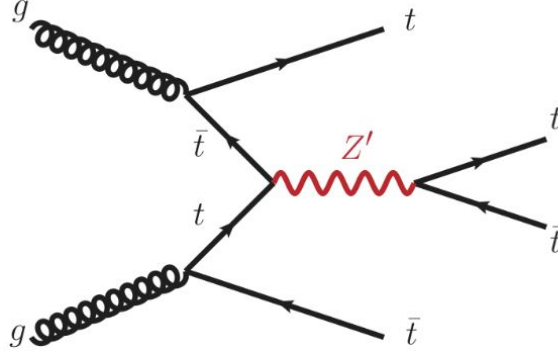


Figure 1: A diagram charting the four top final state

One crucial component of the analysis is the need for precision in the mass resolution. As the accuracy of the mass resolution goes down, potential resonance peaks in the dataset become harder to spot. The mass resolution is shown in figure 2. An increase in accuracy of labelling the resonance and spectator top quarks could greatly increase the precision of the mass resolution. In this report, top quarks are called resonance top quarks if they were one of the two participants in the resonance decay and are called spectators if they did not participate.

The current strategy for labelling the resonance versus the spectators is to look at the momentum of the two particles. As the two resonance tops are highly boosted, they are expected to have much higher momentum than the spectator tops. Thus, the technique was to label the two highest momemta tops as resonance, and the other two as spectators. This method has a decent prediction accuracy; however, there is much room for improvement. A technique that could potentially improve the accuracy of this prediction is to implement machine learning techniques that aim to predict if they are resonance or spectator tops. In this work project we took Monte Carlo data of the four top quarks and applied four different machine learning algorithms to see if this was worth implementing in the analysis.

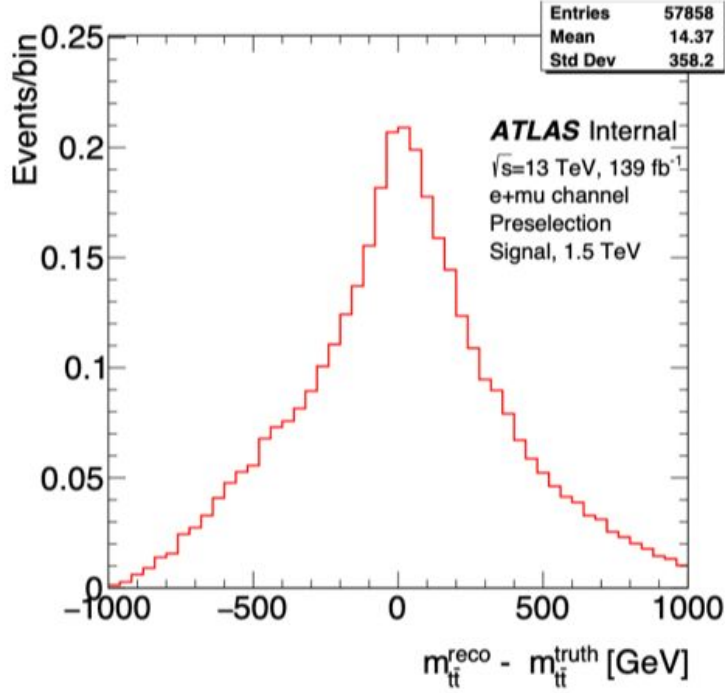


Figure 2: The current mass resonance in the analysis. Graph courtesy of Philipp Gadow

2 Dataset

The Monte Carlo data we used consisted of perfectly reconstructed top quarks. We did not look at what the tops might decay into, nor did we look at information lost in the detector. Applying these four algorithms to more realistic data is a potential next step for this project.

Our dataset was composed of 20,000 events, each with four top quarks. Each top quark is a four vector containing the information about that particle. Also included was truth information as to whether each top was a resonance or a spectator.

From these four vectors, several different quantities can be derived. Two that we looked into were the delta R, or closest distance, value between two top quarks in an event, and the invariant mass of a top quark pair in an event. To calculate this values, we could not use truth information to find the correct pairs as that would defeat the purpose, so instead we choose to label each particle with its minimum delta R value out of all pairings, and its maximum invariant mass out of all pairings. Figure 3 shows the correaltions between all six variables we considered. The diagonal graphs show the comparison of the value for spectators and resonance tops.

78 Through preliminary results we found that the invariant mass was an extremely strong
79 predictor, and the delta R value a rather poor metric. However, after discussion, we
80 decided to exclude these variables from consideration in further machine learning tests.
81 The reason for this decision was because of potential conflicts with the background data.
82 If the machine learning model was trained on invariant mass, it may end up hurting the
overall analysis due to the model being interfered with by the background.

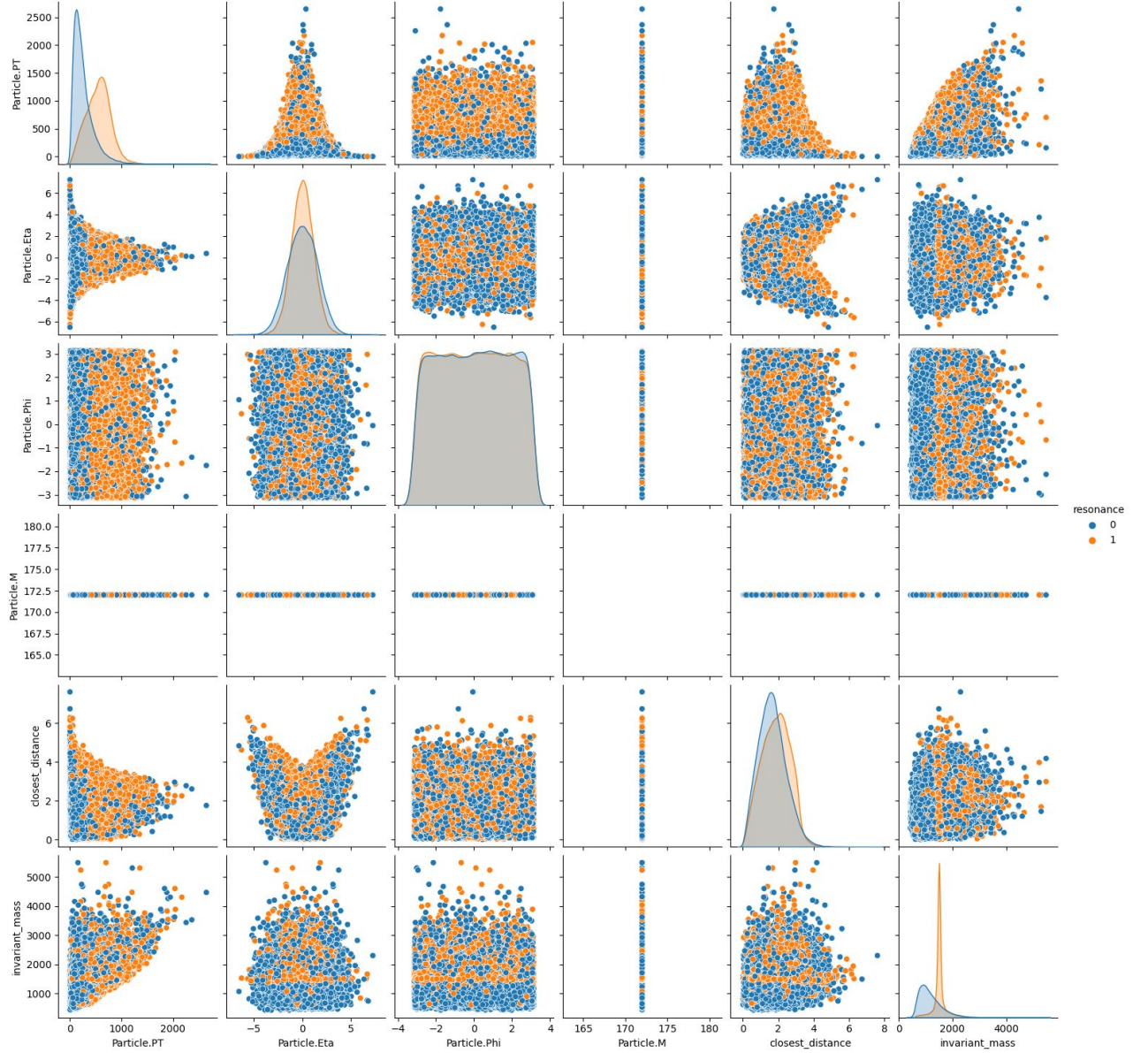


Figure 3: A correaltion chart of all six variables, showing their predictive power

83

3 Methods

During the project we investigated four different machine learning algorithms: k Nearest Neighbors, Boosted Decision Trees, Neural Networks, and Graph Neural Networks. We also looked at the efficiency of a simple cut based approach. Each of the techniques are outlined below.

The main criteria that we used for evaluation was receiver operating characteristic (ROC) curves. These plot the true positive rate versus the false positive rate. Calculating the area under the curve gives a percentage, which we will be using to score the various algorithms. More information can be found in reference [1].

3.1 Simple Cuts

In this approach we made a rough approximation of the cut the actual analysis might make. We cut along the value $pt > 350$, with anything above that being resonance and below it being spectator tops. The ROC curve given from this approach will be used as the baseline, and anything exceeding that efficiency being considered a success.

3.2 KNN

The k Nearest Neighbors algorithm is a useful, simple, and relatively fast machine learning technique. It requires that it is fed a training dataset, which it then stores and uses as a comparison dataset. Whenever it is asked to classify a new event, it calculates the euclidean distance from that event to each separate event in the dataset, then ranks them according to the shortest distance. Finally, it looks at the k nearest values and takes their average, returning that value as its prediction. The value k is given as an input.

Being a relatively simple model, there is less to adjust here than the other methods. The main value to test is how many neighbors to look at. Look at too few, and statistical fluctuations can overly influence results. Too many and the model is not powerful enough to make subtle distinctions between events.

3.3 BDT

Boosted Decision Trees operate by making a branching path that makes different cuts and sorts the event into different classes based on its branch in the tree. It trains on the dataset and produces a tree that at each branch makes a different cut of the data. BDTs are particularly well suited for problems where a lot of weak classifiers are given, as they may be more effective when used together for cuts.

BDTs main feature to adjust is the allowed depth of the tree, but there are a few other less significant features to optimize. The risk here is that a tree that is allowed

to go too deep can overtrain on the dataset. Overtraining is discussed more heavily in section 4.5, along with an example.

3.4 Neural Networks

Neural Networks are the classic example of machine learning. A brief description is that they work by accepting a layer of input features, then connecting these to unspecified neurons in several hidden layers, and finally outputting to your output features. This method works by using training to adjust the weights of the connections from each layer to find an optimized set of weights between neurons that allows the model to most accurately predict correctly.

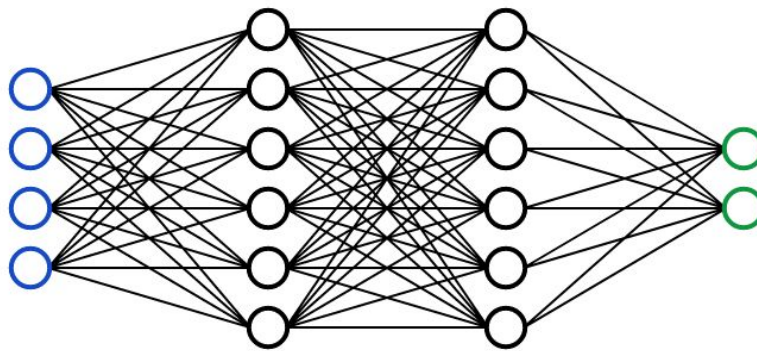


Figure 4: An example of how a neural network with two hidden layers might look

For neural networks, you can adjust the number of hidden layers and the amount of neurons in each, but there are other parameters to adjust as well. These include how many events per training batch, the total allowed training time (training covers all events in one "epoch") and what loss function is used to calculate the difference between predicted and truth.

3.5 Graph Neural Networks

Graph neural networks are simply neural networks that are built to accomodate graph structure in the problem. Despite this simple description, they are quite complex in nature. There are three main problem formulations: node classification, edge classification, and graph classification. We tried all of these approaches before settling on node classification, although edge classification seems to also fit our problem rather well. More work could be done on determining which approach would work best. Reference [2] goes into depth on graph neural networks in particle physics.

The graph neural network implementation we used is called GraphSAGE. The paper in reference 3 explains it in detail.

4 Measurement

We will show the various methods in comparison with one another. Only the first three ROC curves will be shown in the interest of space, and the area under the curve will be given instead in Table 1.

Table 1: Values for the AUC for each method

Method	AUC
Simple Cuts	0.78
KNN	0.85
BDT	0.86
NN	0.87
GNN	0.82

4.1 Simple Cuts

This approach will serve as the baseline value. All four of the machine learning approaches managed to exceed the values for this cut.

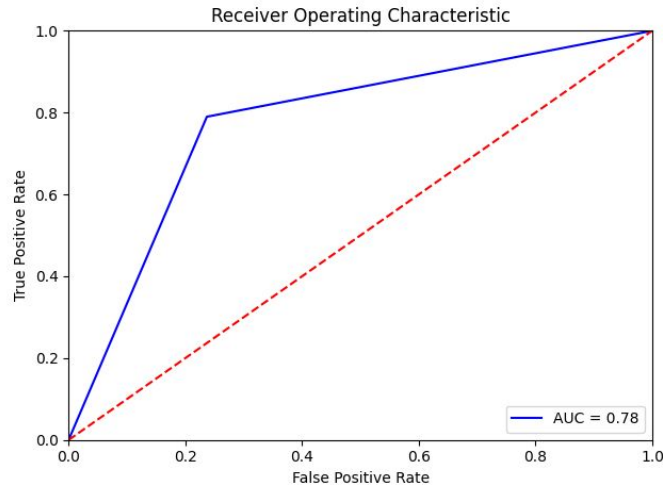


Figure 5: The ROC for the simple cuts

4.2 KNN

For the k value, we ended up using a value of 400 neighbors. We found any more than this and the ROC began to fall. Some testing could be done to pick a more precise value or to see if this value holds for larger datasets, but for this method, there is relatively little optimization to do. It is interesting to note that the KNN method managed to beat the human cuts approach after very little time spent implementing and optimizing.

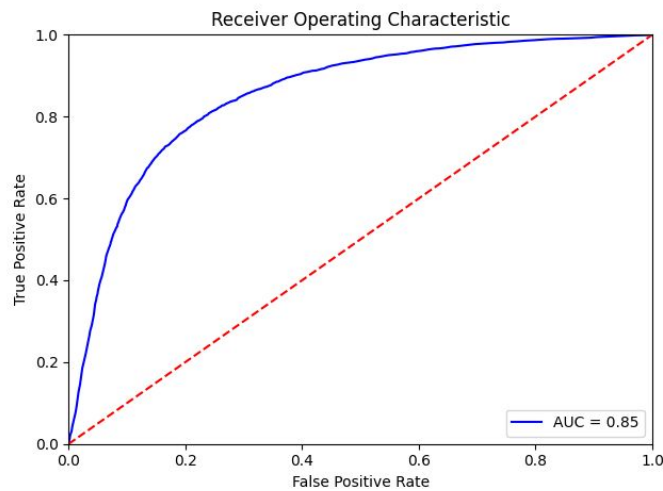


Figure 6: The ROC for the KNN method

4.3 BDT

For the boosted decision trees, we allowed for a max of 5 levels in the tree. For this method, the other parameters besides tree depth were not optimized much. More work could be done to find an exact optimization for BDTs in this problem. Considering the fact that they were the second highest approach, it may be worth it to investigate this method further.

4.4 Neural Networks

The neural network performance was the best out of all techniques looked at. For this model, we settled on 32 hidden nodes per layer with two hidden layers. One challenge for working on NN is the difficulty in selecting these parameters. There were many values left untested that may work better. More hyper parameter optimization could be done to likely improve the values.

Shown in figure 8 is a graph of the loss versus the epochs. This graph shows that

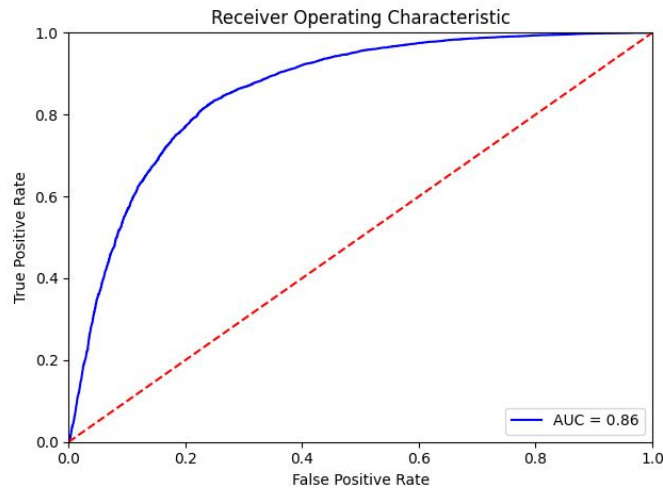


Figure 7: The ROC for the BDT

175 the neural network reached its lowest loss value fairly quickly into the training. This
 176 suggests that a shorter time could be used for training.

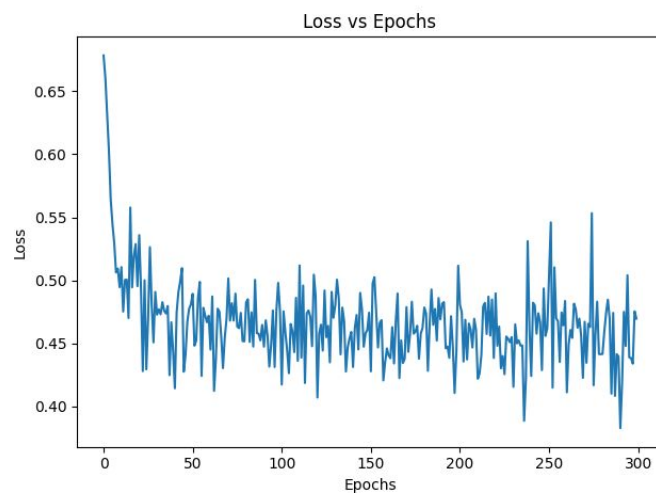


Figure 8: A graph of Loss vs Epochs for the Neural Network

176

177 4.5 Graph Neural Networks

178 We tested multiple different configurations of GNNS before deciding on GraphSAGE. In
 179 the implementation we selected, it shared all the features that could be changed as a NN,
 180 but it also had several independent features. One changing factor was the aggregation

function used. We tested four different aggregation functions, the simplest being the mean.

Figures 9 and 10 show the accuracy versus epochs for two different cases. Figure 9 is uses one hidden layer and the mean aggregator function. Figure 10, despite appearing to rapidly rise in accuracy, performed much worse than the case in Figure 9. This poor performance was due to overtraining. The 2nd, more advanced, case began to overtrain on the training dataset when left running long past it reaching its optimal point. It started to compensate for small features in the training data, which in turn made it worse for modeling the test dataset. The accuracy vs epochs plot for the one hidden layer approach. This set up scored 0.82, whereas the other had a score of 0.77, worse than the simple cuts.

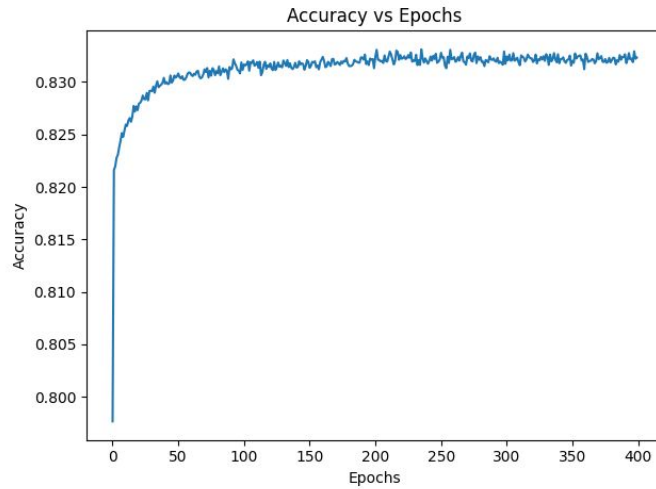


Figure 9: The accuracy vs epochs plot for the one hidden layer approach. This scored 0.82

5 Conclusions

After working with four machine learning methods, it seems likely that machine learning would be helpful for increasing the accuracy of the mass resolution. The best performing algorithm were neural networks, but it seems that graph neural networks still have much room for optimization due to their complexity. There remains much work left to be done by reading more about graph neural networks and finding a particular model and problem formulation that fits our case best.

The next steps for this project would be to begin testing these algorithms on more complex, realistic simulated data that included non-fully reconstructed particles. After

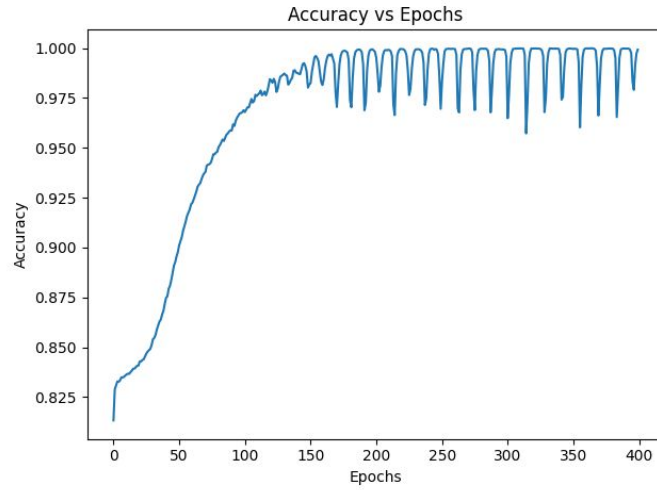


Figure 10: The accuracy vs epochs plot for the three hidden layer approach and complex aggregation function. This model scored 0.77, despite appearing to be much better than the previous results

203 that, it would be helpful to construct a projected difference in the mass resolution to
 204 see if these techniques would yield significant improvement for the analysis.

References

- [1] Bewick, V., Cheek, L. & Ball, J. Statistics review 13: Receiver operating characteristic curves. Crit Care 8, 508 (2004). <https://doi.org/10.1186/cc3000>
- [2] Jonathan Shlomi, Peter Battaglia, Jean-Roch Vlimant. "Graph Neural Networks in Particle Physics" Machine Learning: Science and Technology, 2 (2021) <https://arxiv.org/abs/2007.13681>
- [3] William L. Hamilton, Rex Ying, Jure Leskovec. "Inductive Representation Learning on Large Graphs" <https://arxiv.org/abs/1706.02216>