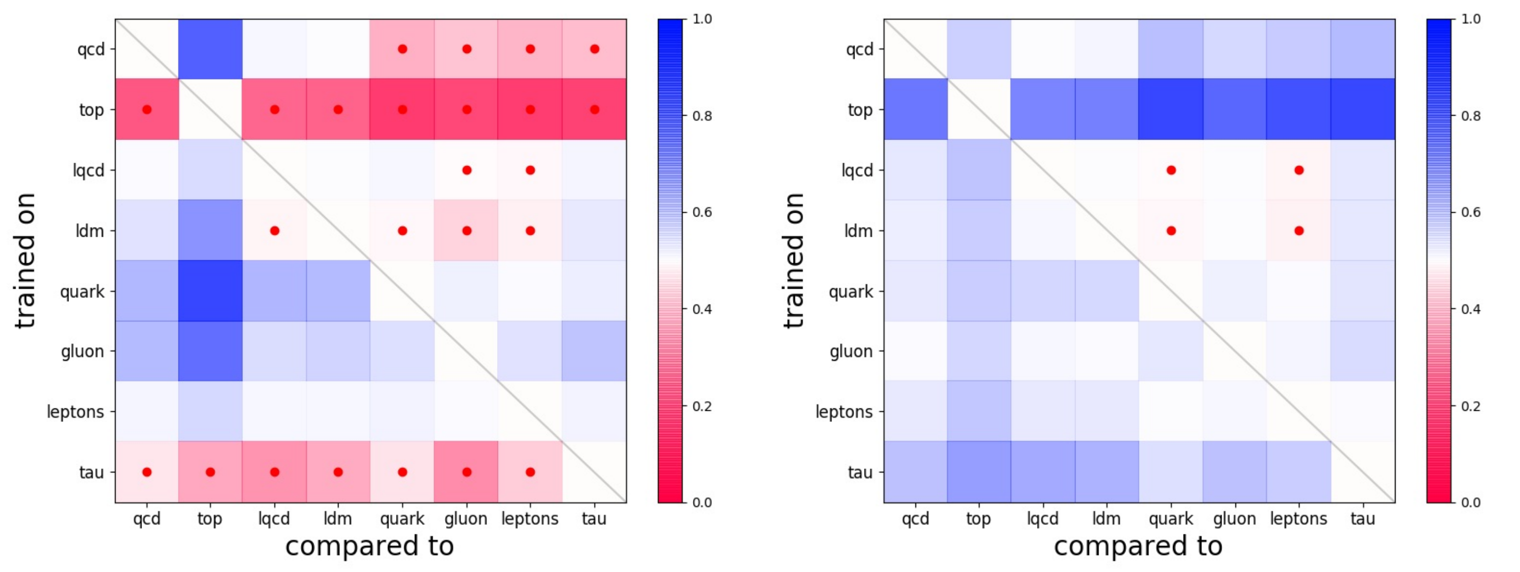
[Back to CV]

[Download thesis]

In my master thesis I tried to use machine learning to find new physics. I did this, by using methods from anomaly detection, namely autoencoder.
These autoencoder are trained to reproduce any jet you input into them, so jets that are anomalous, like those that are produced by any non standard model physics, are reproduced worse, which means, that you can use the loss of the autoencoder to seperate anomalies.
The initial focus of my master thesis, was to combine these autoencoder method with graph like neuronal networks and see if this helps their accuracy.
Even though I was able to combine those two ideas into working graph autoencoder, using them to find new physics jets is a little more difficult: The best networks are those only utilizing a few particles, and those that are not able to reconstruct angles at all.
This is because of the way these networks are usually evaluated: Instead of simulating some theory, we try to find those jets that are produced by top quarks in a background of jets that are produced by the rest of qcd.
I was able to notice, that a difference between qcd and top jets, namely the width in eta/phi space, is exactly what the autoencoder focuses on.
This trivial feature is able to reach a very good seperation, and even a badly trained model is sensitive to it, but sadly you have to conclude that it is fairly useless for detecting new physics, since the only new anomaly these networks would be able to find are those that are even wider.

So to create a network that is able to find new physics, I needed to make it ignore this feature.
I tried two different approaches, first one using a normalization to remove the width from our input features and another extending the power of the autoencoder by my own anomaly detection algorithm, I call oneoff networks.
These oneoff networks are based on an feature in the training of normalized networks and allow me to create anomaly detectors that are very general.
The price for this generality is, that quality of the difference (in the top qcd case) is a bit lower.
You can see the benefit very clearly in the following plot



In these plots I show 8 data sets, being compared to each other.
This means, instead of measuring how well a model can do the task of finding a specific jet type, I use 56 comparisons.
Each colored square is one comparison and should be deeply blue for a perfect separation.
It is white if the network does not find any difference and red when this difference is negative.
On the left you see the result for a simple dense autoencoder, while the much more blue version on the right represents my graph autoencoder with normalization and oneoff networks.

Finally I uploaded my graph autoencoder code to pypi and wrote a documentationfor it at [grapa.readthedocs.io].