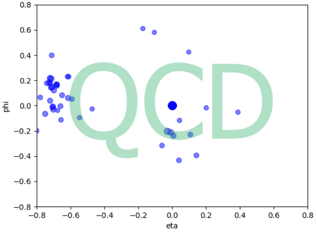
Jets, machine learning and new physics mining

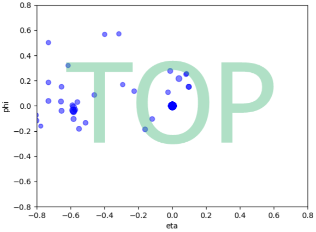
Simon Kluettermann

9. März 2021

Top Tagging

- Looking at LHC jets (particle clouds)
- Millions of jets, maybe some of them weird in some way
- But at most a few and in some completely unknown way
- So use ML to find them
- Usually tested by trying to find top jets in a qcd background





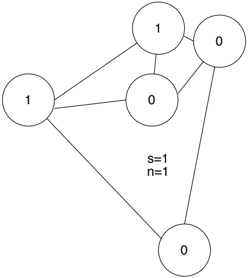
Idea

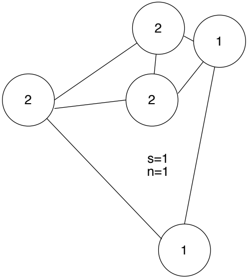
- Qcd Or What? (arxiv:1808.08979)
  - unsupervised $\Rightarrow$ so can find anything abnormal
  - Convolutional Networks
- ParticleNet (arxiv:1902.08570)
  - supervised $\Rightarrow$ so can only find top jets
  - Graph Neuronal Networks

- Combine ideas from both
- Into a Graph Autoencoder

What are Graph Neuronal Networks?

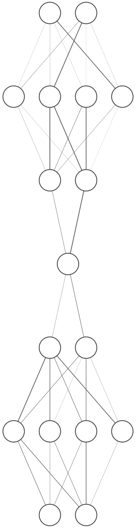
- Graph: objects with relations between them.
- Update each object by their neighbours.
- Make this learnable and stack it $\Rightarrow$ Graph neuronal network
- $x_i^{t+1} = A_i^j \cdot n \cdot x_j^t + s \cdot x_i^t$
    - A: Adjacency matrix, defines the graph
    - x: Feature vector, defines each object
    - s: Self interaction matrix
    - n: Neighbour interaction matrix





What are Autoencoder?

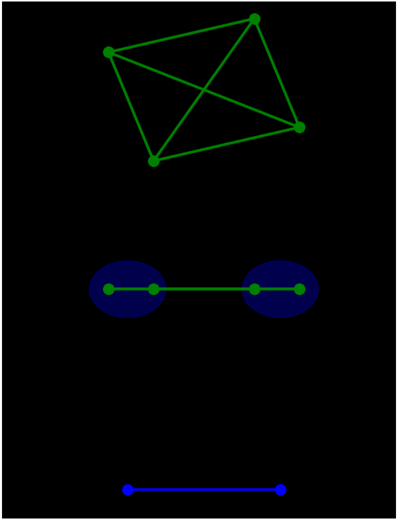
- Compress data into a latent space (lower size)
- Decompress it again ⇒ loss is difference to input
- loss only small for training data
- if you input anomalous data, the output wont be anomalous⇒the loss is high
- loss: measure for how anomalous an input is

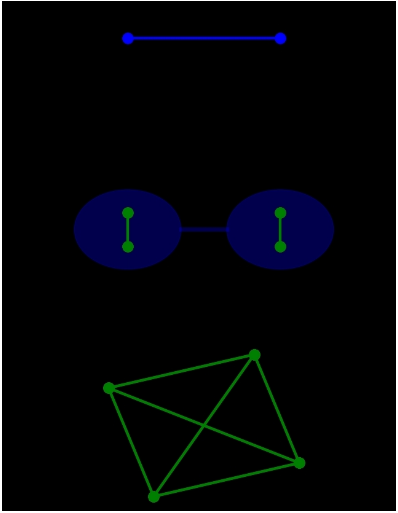


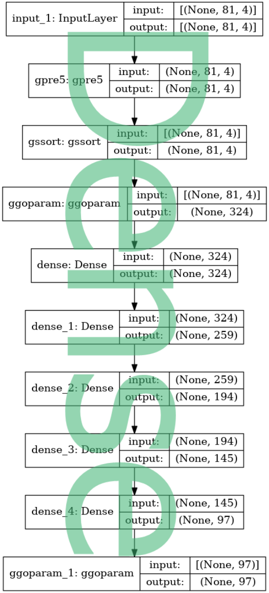
Literature

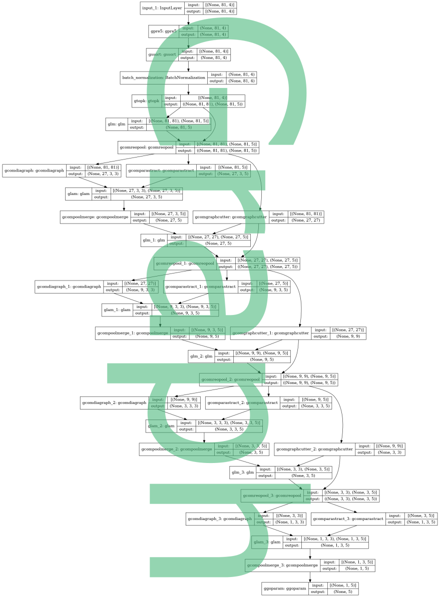
- Need some function to go from a big graph to a small one
- similar to a Pooling operation for a convolutional network
- Seems simple enough but if you look at the literature
  - slow...and the benefits...are less clear (arXiv:1907.09000)
  - advance...has lagged behind (arXiv:1907.00481)
  - one cannot simply pool ... (arXiv:1806.08804)

Implementation






input_1: InputLayer
input: [(None, 81, 4)]
output: [(None, 81, 4)]
gpre5: gpre5
input: (None, 81, 4)
output: (None, 81, 4)
gssort: gssort
input: [(None, 81, 4)]
output: (None, 81, 4)
ggoparam: ggoparam
input: [(None, 81, 4)]
output: (None, 324)
dense: Dense
input: (None, 324)
output: (None, 324)
dense_1: Dense
input: (None, 324)
output: (None, 259)
dense_2: Dense
input: (None, 259)
output: (None, 194)
dense_3: Dense
input: (None, 194)
output: (None, 145)
dense_4: Dense
input: (None, 145)
output: (None, 97)
ggoparam_1: ggoparam
input: [(None, 97)]
output: (None, 97)



Assuming

The higher the loss of the comparison network

The more likely this is a top jet.

You reach an AUC of

0.908

- Already fairly satisfied
- If I beat this, my graph networks work

- AUC score
- between 0 and 1, higher=better
- 1:perfect,0.5:random

Assuming

The higher the loss of my graph network

The more likely this is a top jet.

You reach an AUC of

0.910

- So my network works
- But...

- AUC score
- between 0 and 1, higher=better
- 1:perfect,0.5:random

Assuming

The higher R**2

The more likely this is a top jet.

You reach an AUC of

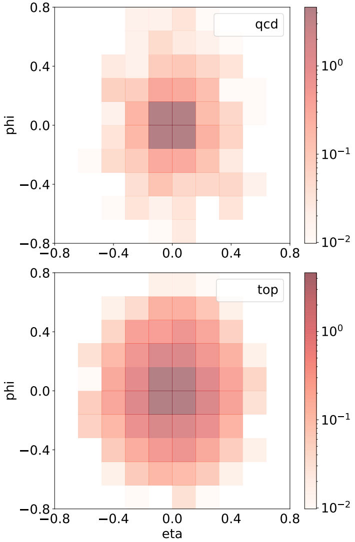
0.915

- trivial network
- best score yet

- AUC score
- between 0 and 1, higher=better
- 1:perfect,0.5:random

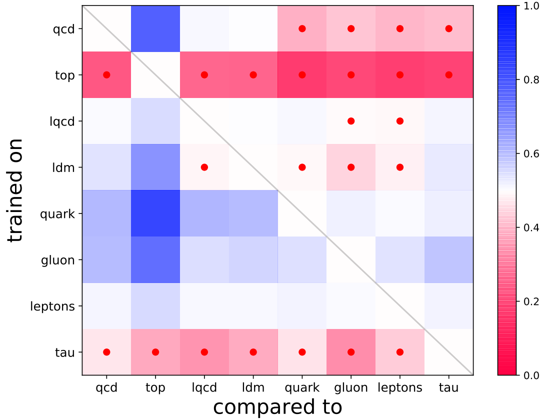
How to explain this?

- trivial difference between qcd and top jets
  - top jets are generally wider
- untrained ae ⇒ loss is width
- more or less useless, since this can only find top (wide) jets

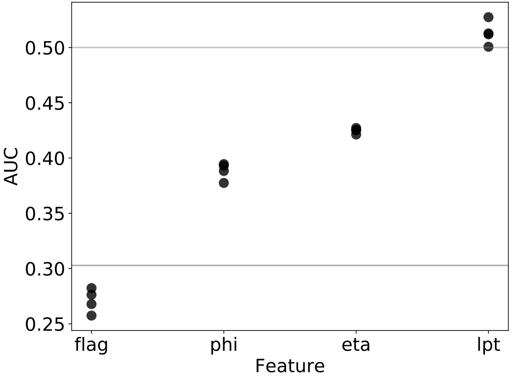


Other data

- The more blue the better, and if a pixel is red (dot) it is not detectable
- For the comparison work only 10(32)/56 are detectable
- useless, except for qcd vs top

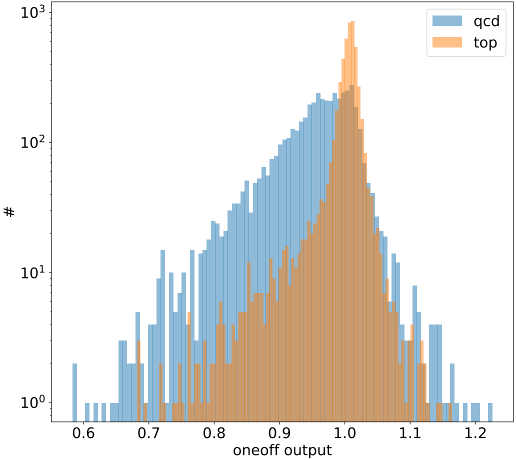


- How to solve this?
- First idea: normalisation
- This works, but again a feature in data
- Luckely this time really useful
- flag is a constant, but more decisive than the rest



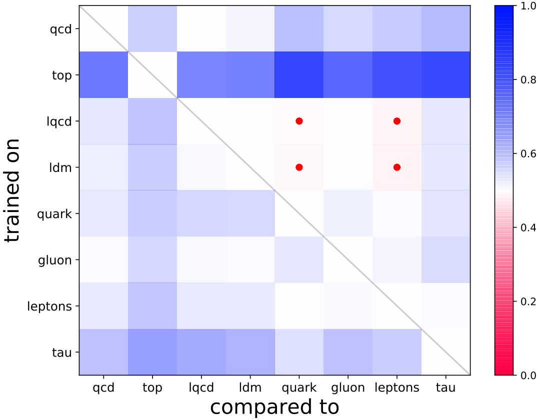
OneOff Networks

- So use this:
- Train a network to output a constant
- $loss = (f(x) - 1)^2$
- Anomalous data usually does not reproduce the same constant (as good)



Results

- The more blue the better, and if a pixel is red (dot) it is not detectable
- Quality is not final, but
- here do 48(52)/56 comparisons work



Conclusion

- Anomaly detection migth be useful in jet physics, but can go wrong too
- Graph neuronal networks allow you to better understand how your network works
  - use my graph autoencoder : grapa.readthedocs.io
- General Anomaly detection requires a lot of care
- OneOff Networks are (at least) a good try
- Probably countless applications other than just jet classification