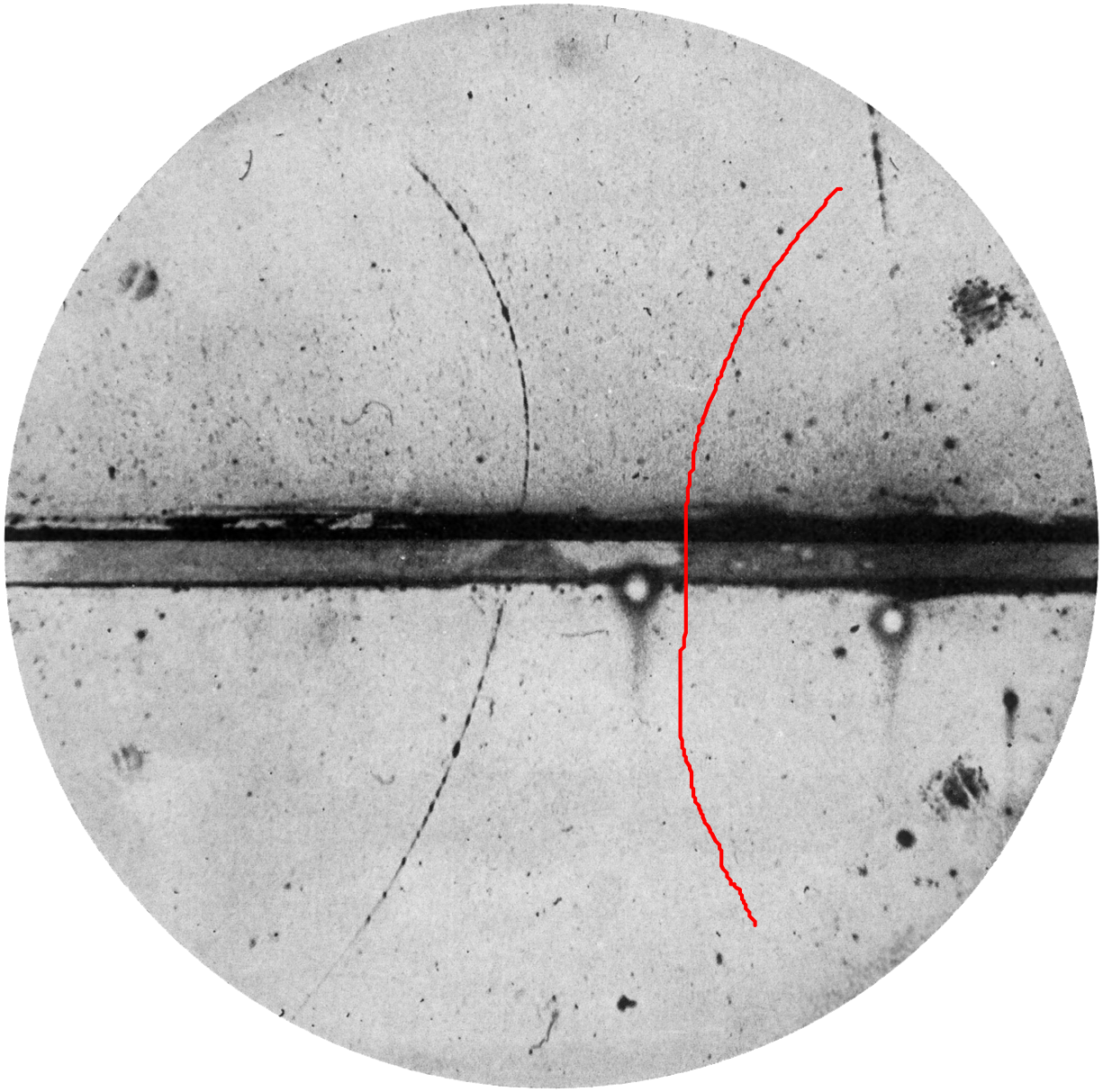
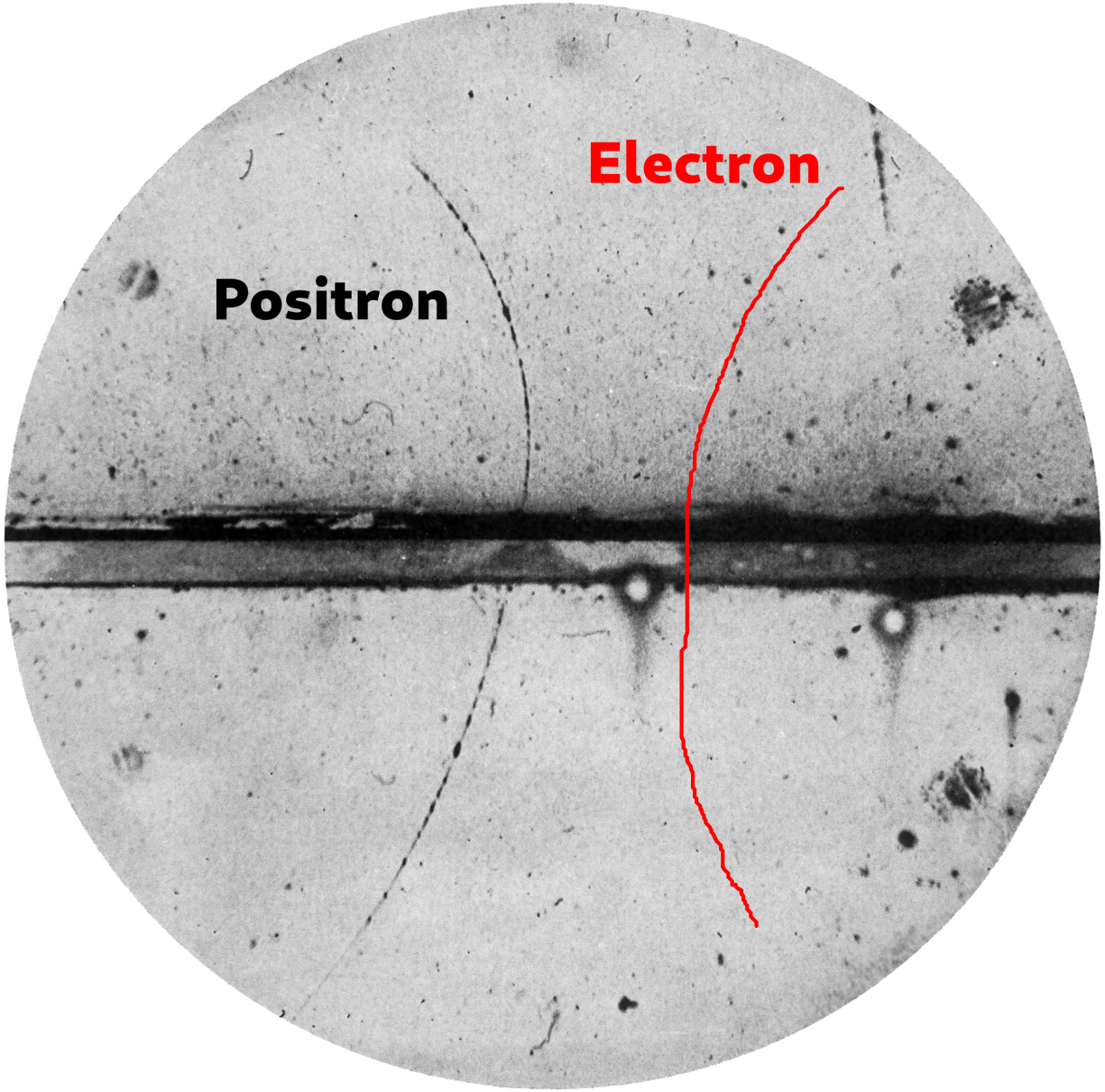
Anomaly detection in physics

Simon Kluettermann

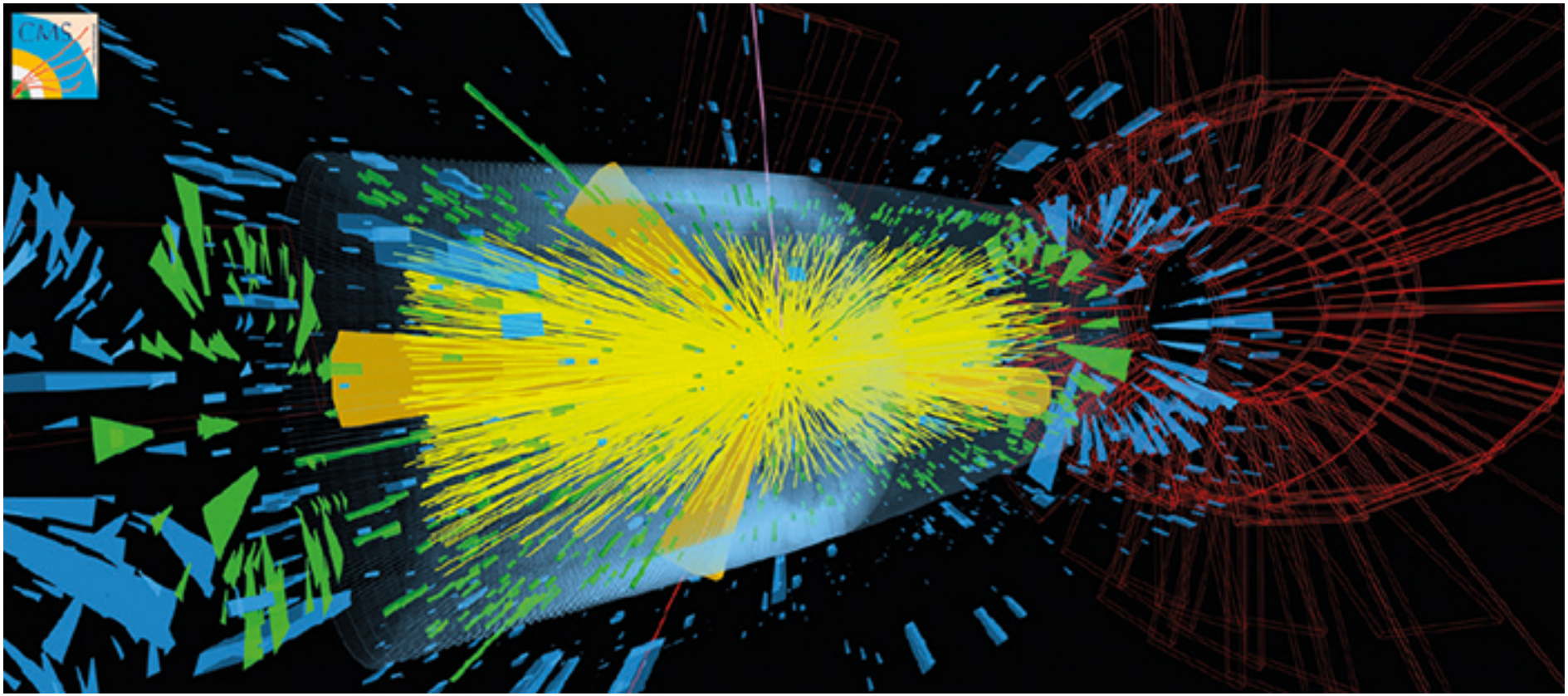
8. Juni 2021




Electron
Positron

- Detection of Antimatter
- Not extremely hard to detect, but deserved Nobel Prize
- Today:
  - fundamental building block of physics
  - required to understand advanced chemistry/biology
  - medical uses
- Also Today: Anomalies are sorely needed in Physics

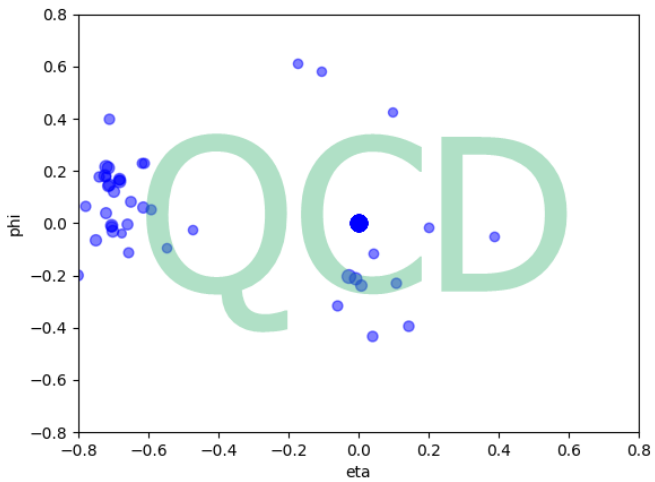
Instead of 1000 Events/day⇒1 Petabyte/day

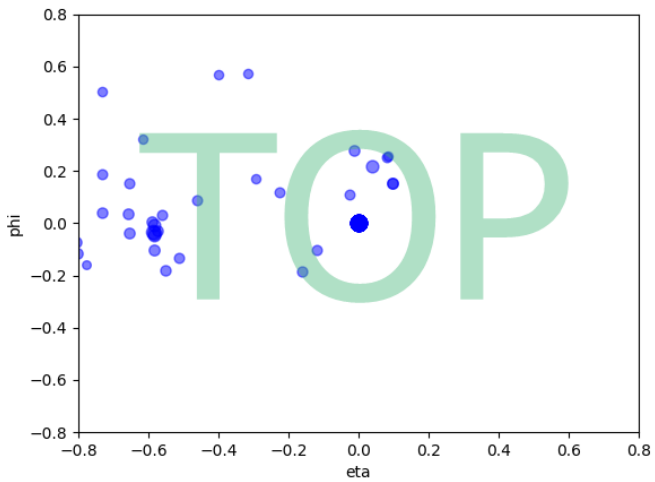


⇒require computers

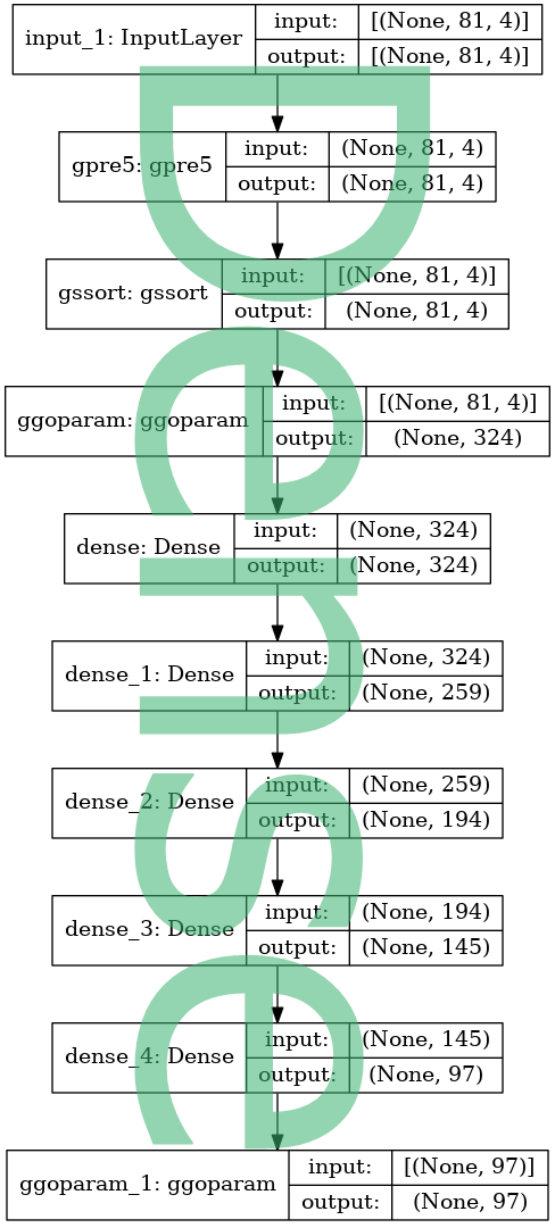
Top Tagging

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



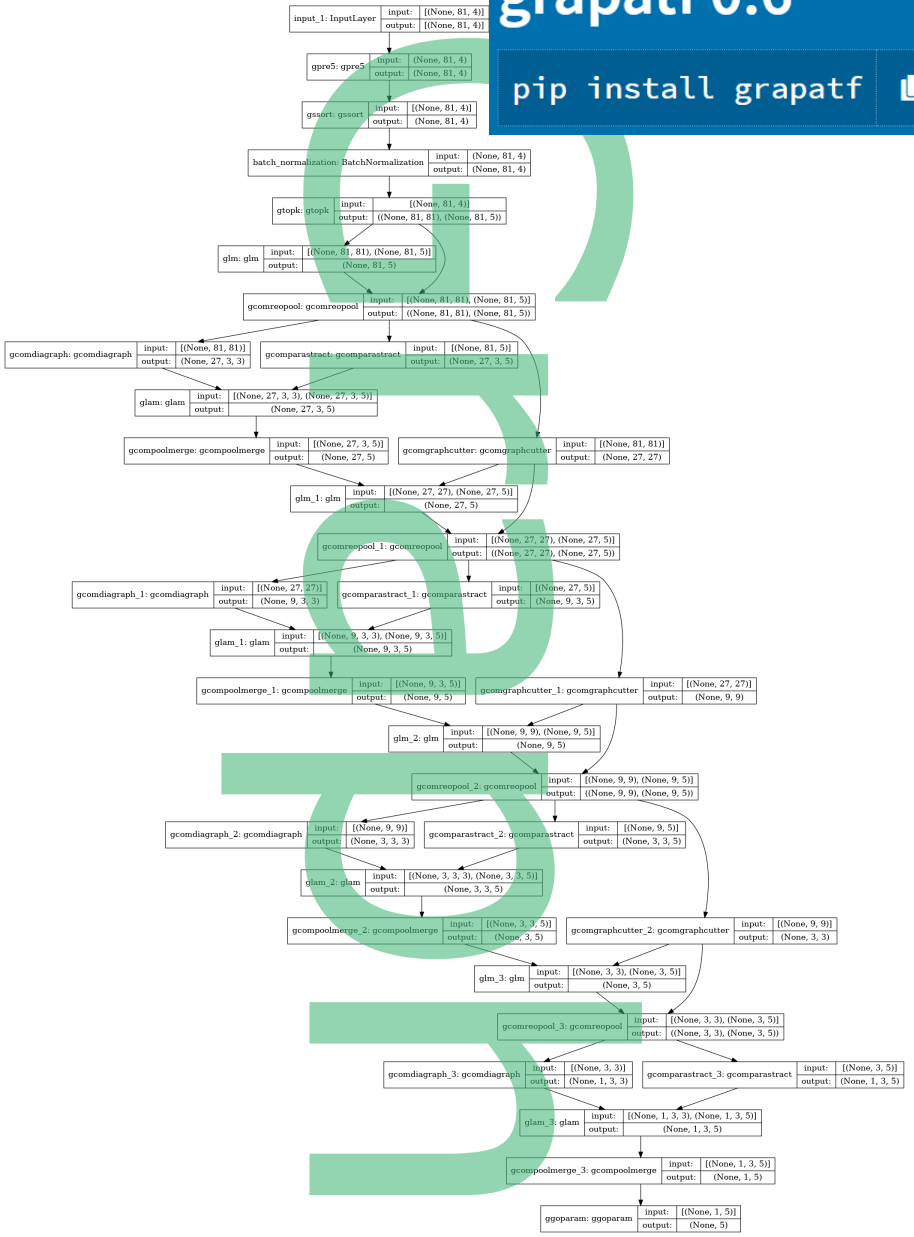


- ParticleNet
  - supervised
  - Graph Neuronal Networks
- Qcd Or What?
  - unsupervised
  - Convolutional Networks
- Combine ideas from both
- Into a Graph Autoencoder



grapatf 0.6

```
pip install grapatf
```



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

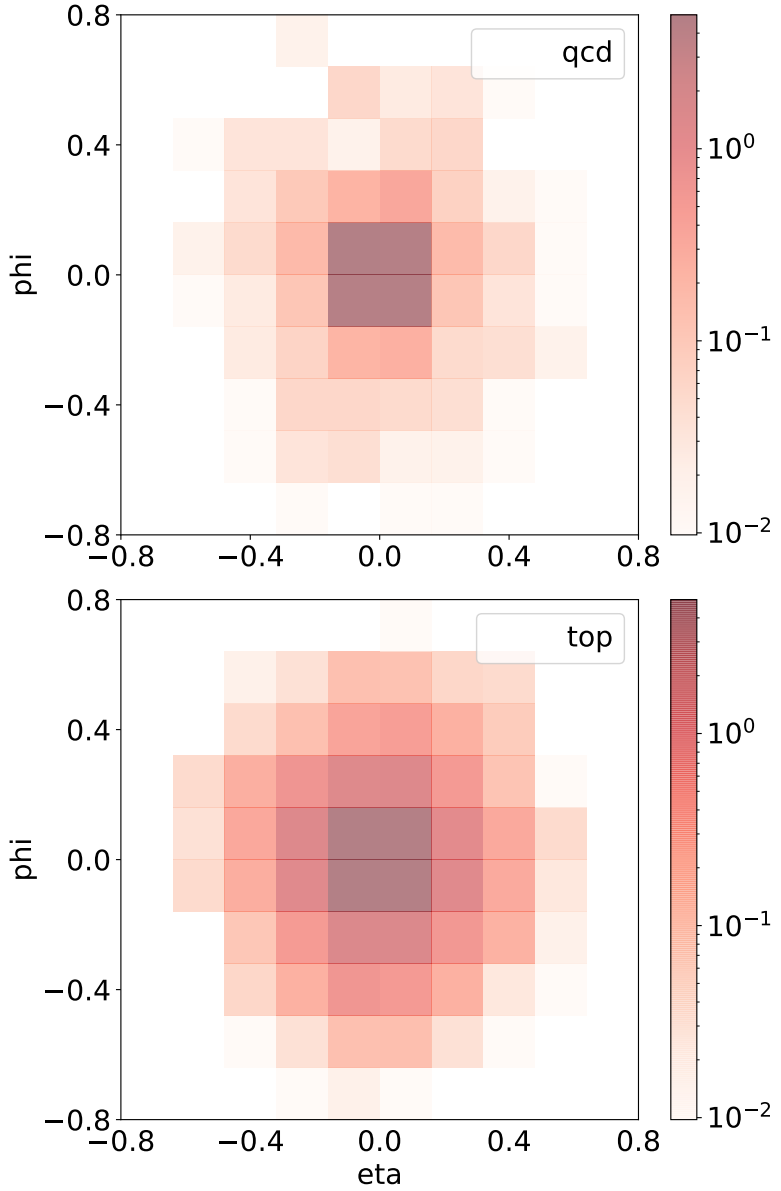
0.915

- trivial network
- best score yet

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

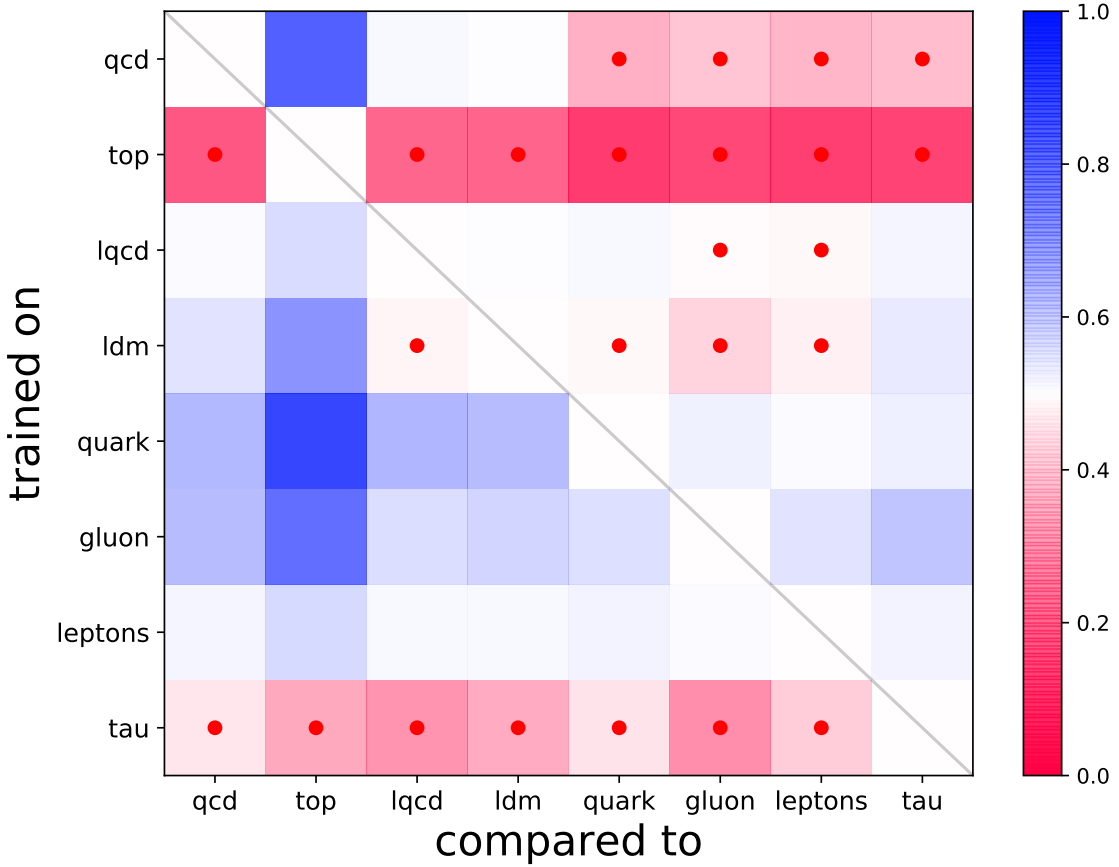
Results

- Why? Trivial difference between the jets we look at
- Width is measured by bad autoencoders
- Ae does the same as the radius but worse



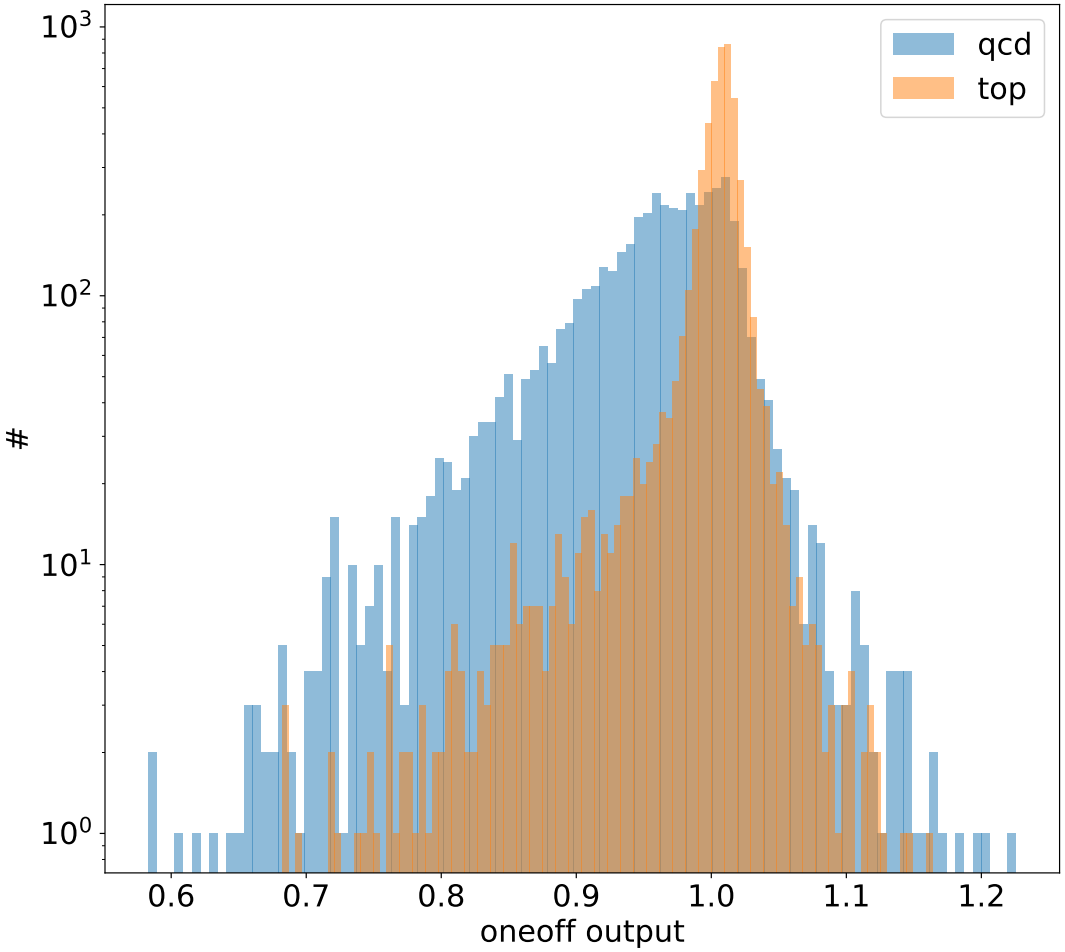
Results

- 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



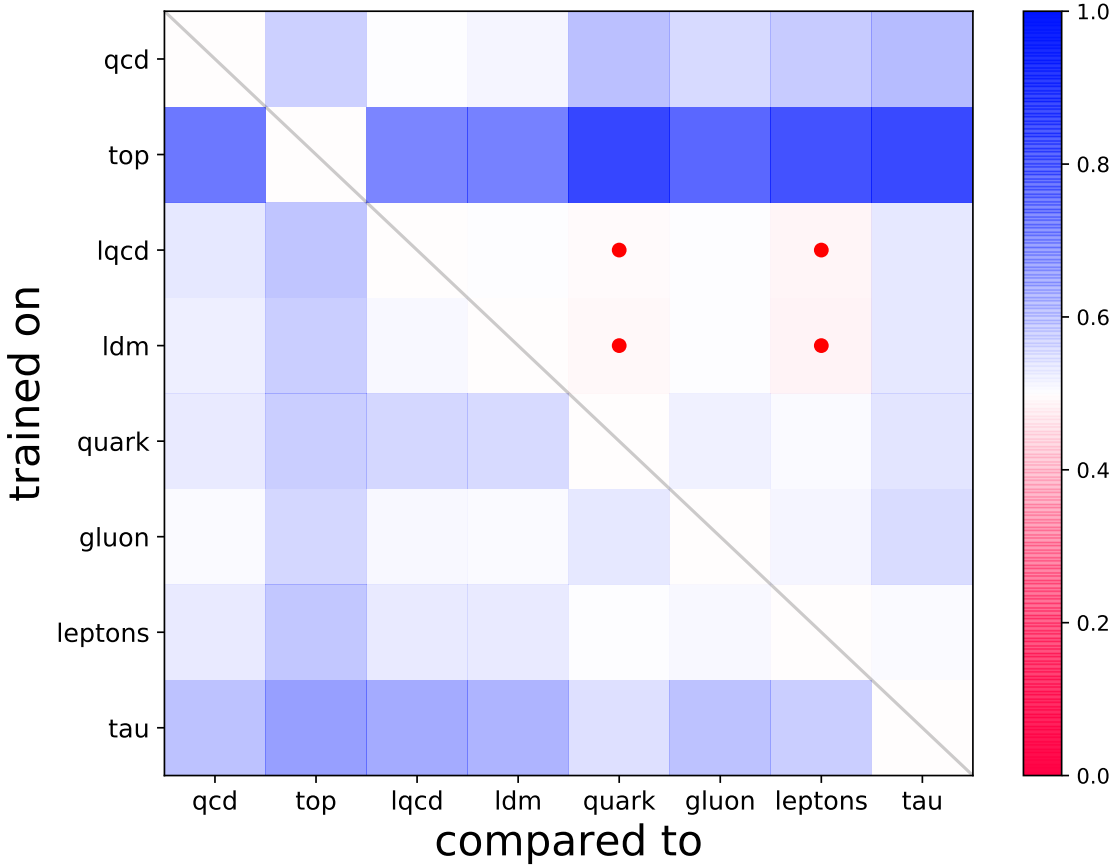
OneOff Networks

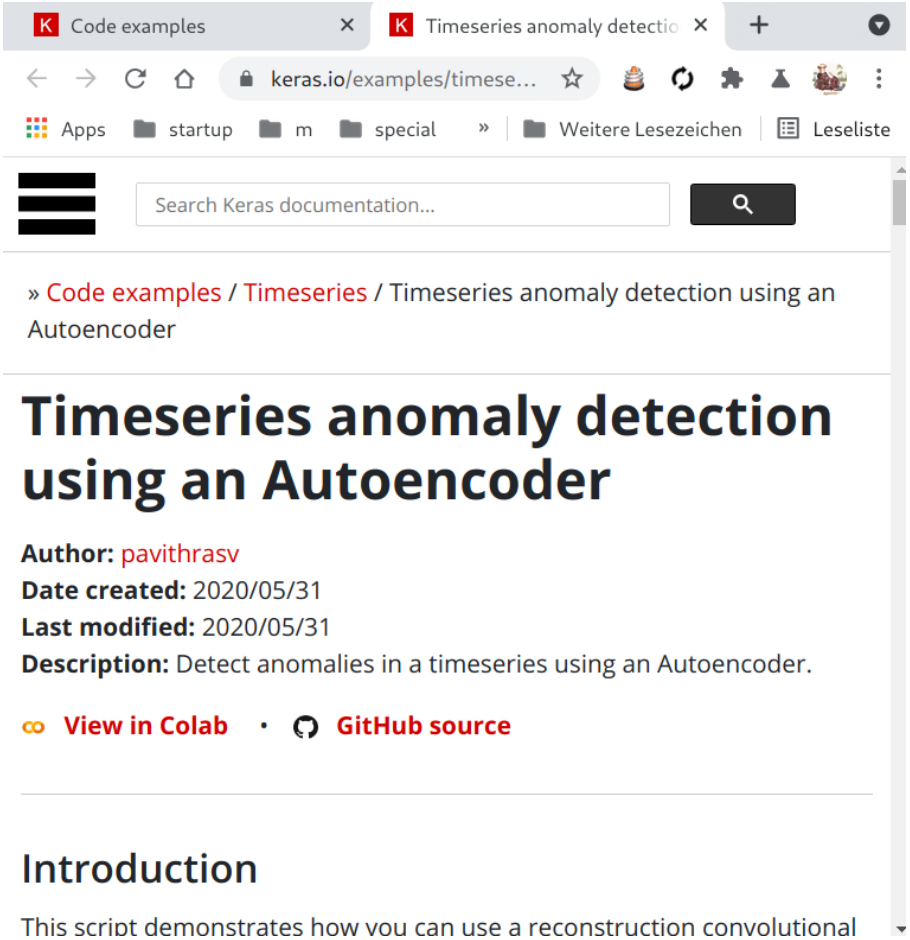
- So improve it using my own algorithm:
- Train a network to output a constant
- $loss = (f(x) - 1)^2$
- Anomalous data usually does not reproduce the same constant



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





	Autoencoder	OneOff
time	123s	22s
AUC	99.6%	99.92%
$1 \cdot \frac{1}{1-AUC}$	260	1290

Will be used for the rest of this Presentation

Orthogonal Anomaly Detection

- Usually nd input $\Rightarrow$ 1d anomaly score
- Why not nd input $\Rightarrow$ md anomaly score
- Retraining an anomaly detection algorithm provides about the same result
- Can you change this?
- And so create an multi-dimensional output anomaly detection algorithm
- $\Rightarrow$yes

- Create a general Framework for orthogonal anomaly detection
- Migth improve Generality, Interpretability and general Strength of anomaly detection

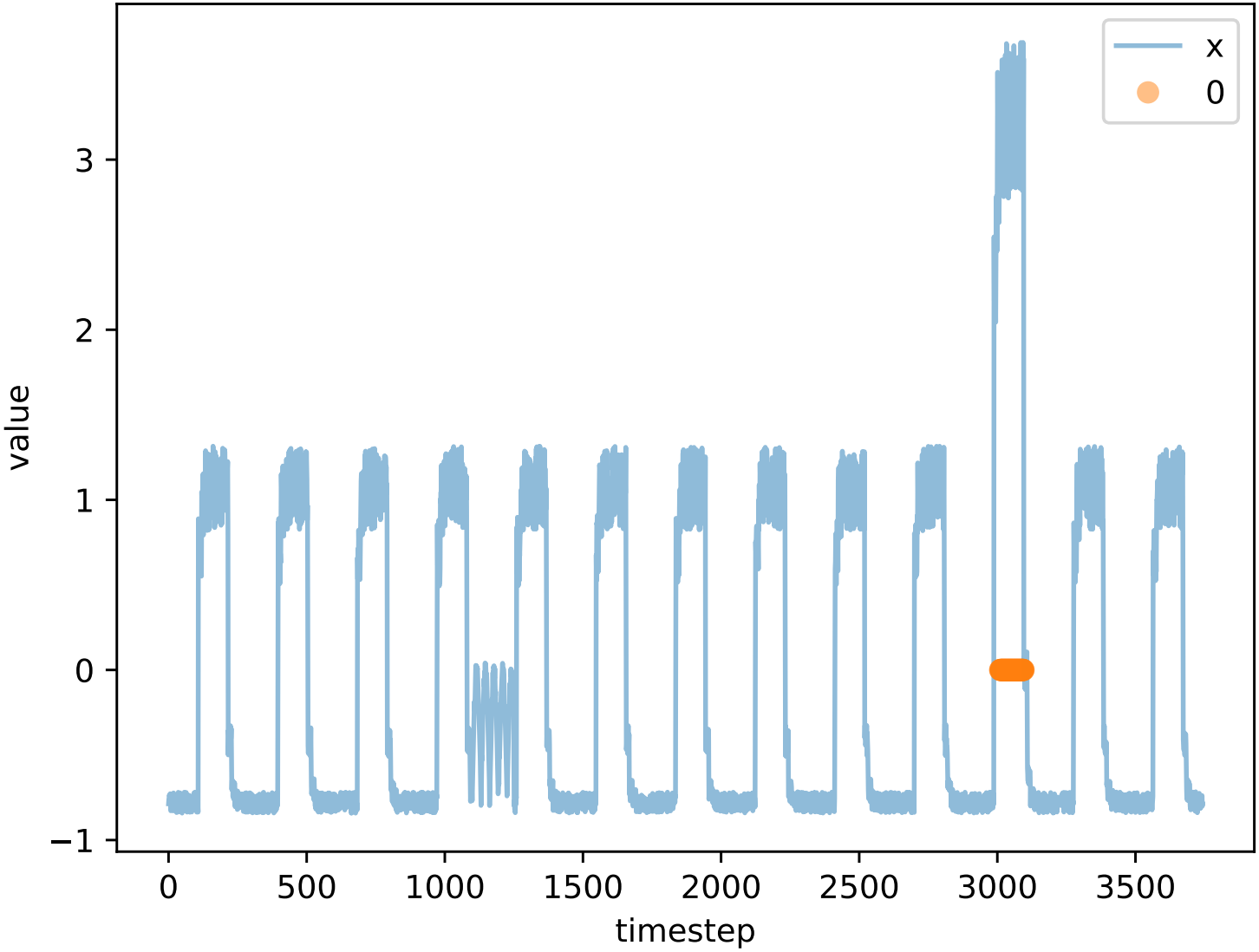
How

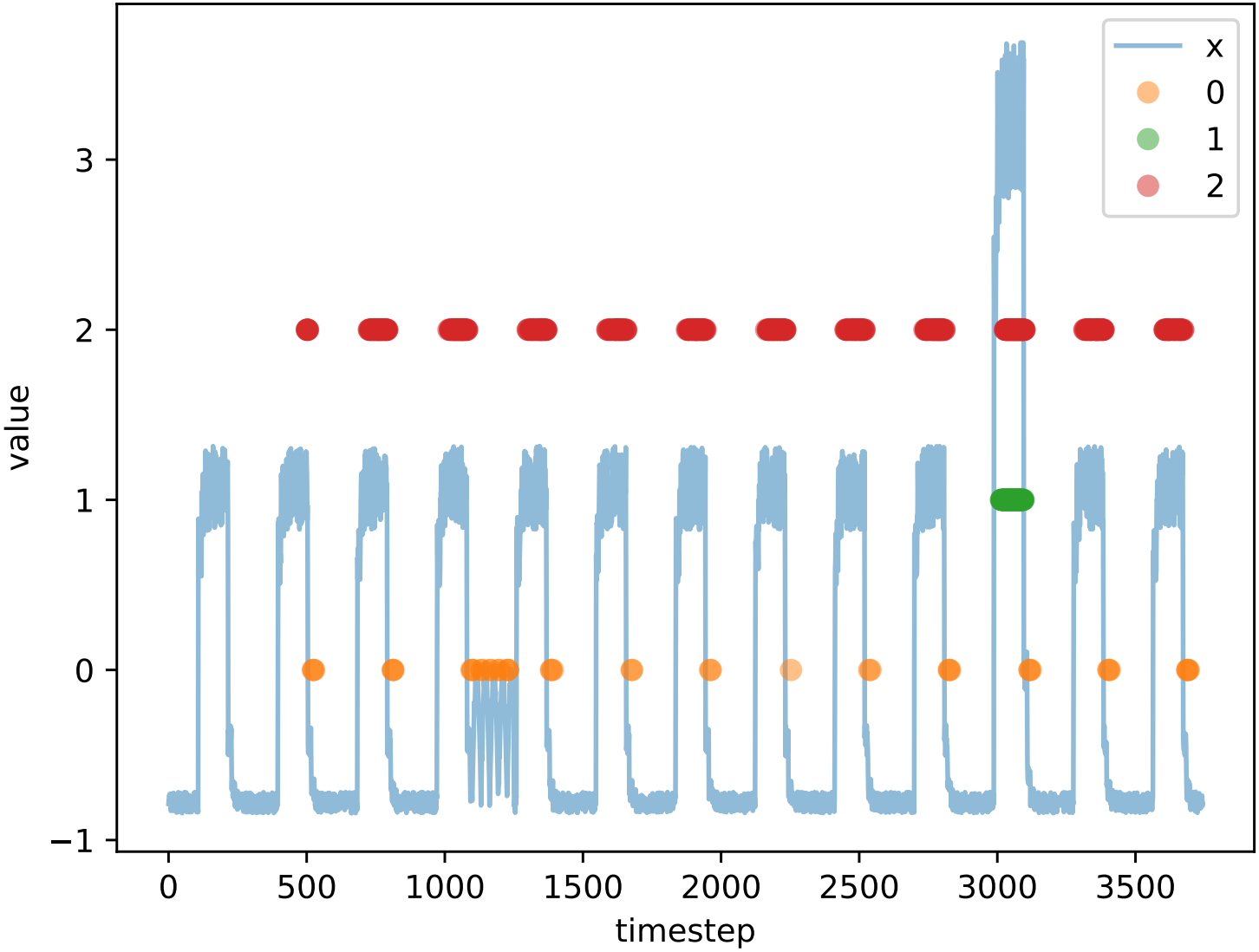
- Chance loss function
- From $loss = (f(x) - 1)^2$
- Too $\sum_{i<j} |(\mathrm{f_i}(x) - 1) \cdot (\mathrm{f_j}(x) - 1)|$

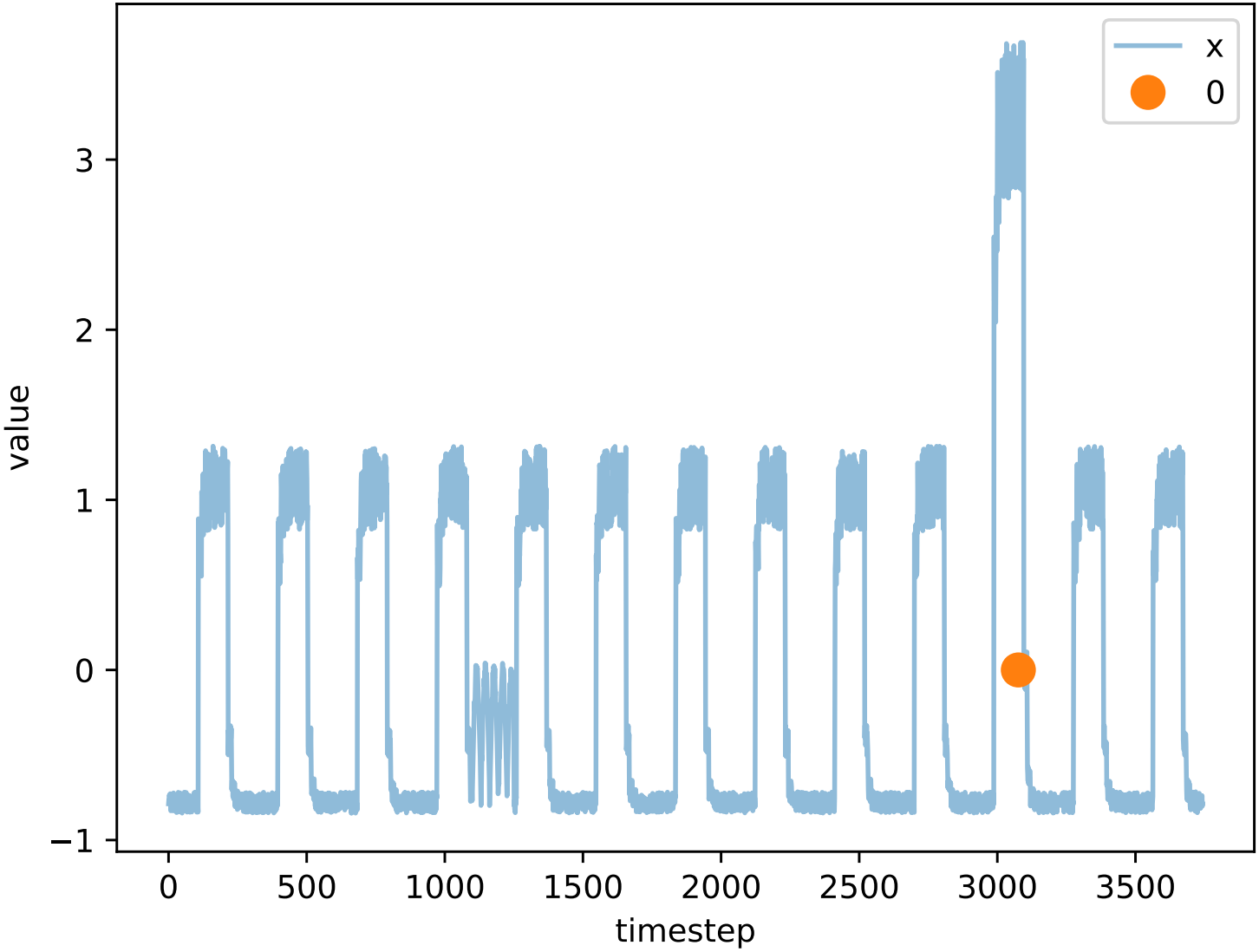
$$\begin{bmatrix} 1 & 0.032 & -0.015 \\ 0.032 & 1 & -0.14 \\ -0.015 & -0.14 & 1 \end{bmatrix}$$

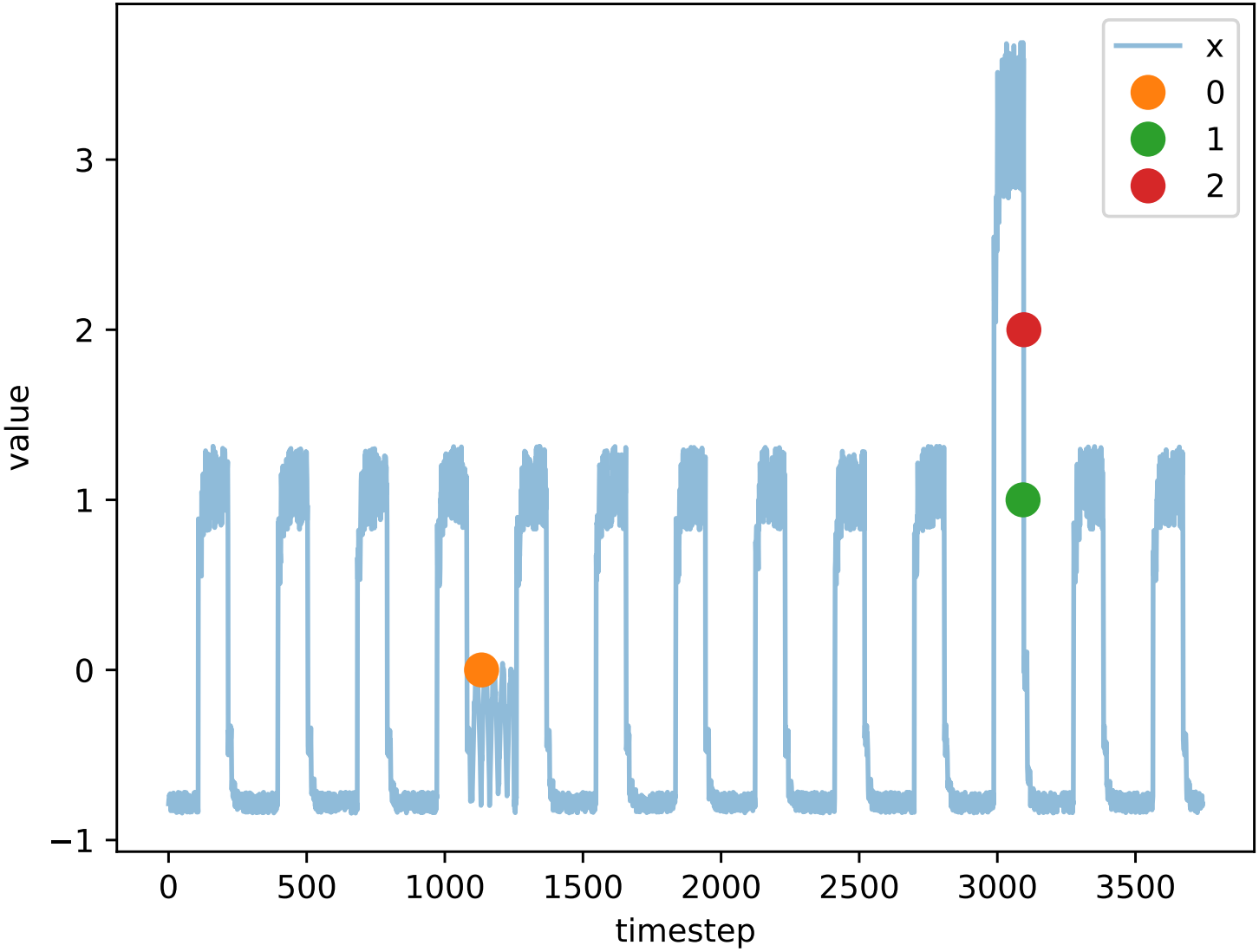
Generality

- Less important features can go hidden
- Also sometimes 2 errors can cancel each other out
- By having multiple outputs, you can reduce these effects
- Here I look at a different project, focussing on recurrent anomaly detection
- Same dataset as the keras example, but with another more complicated anomaly


x
0
value
timestep
3
2
1
0
−1
0
500
1000
1500
2000
2500
3000
3500

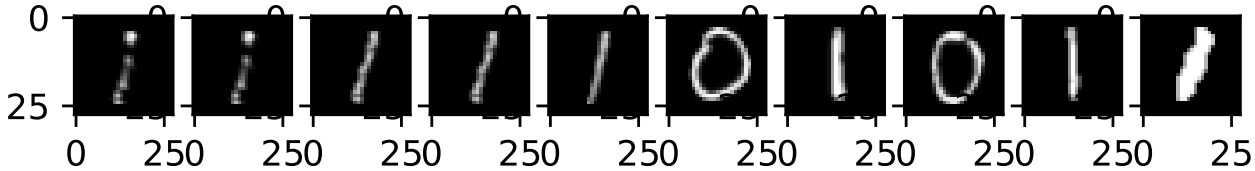

x
0
1
2
value
timestep
3
2
1
0
−1
0
500
1000
1500
2000
2500
3000
3500

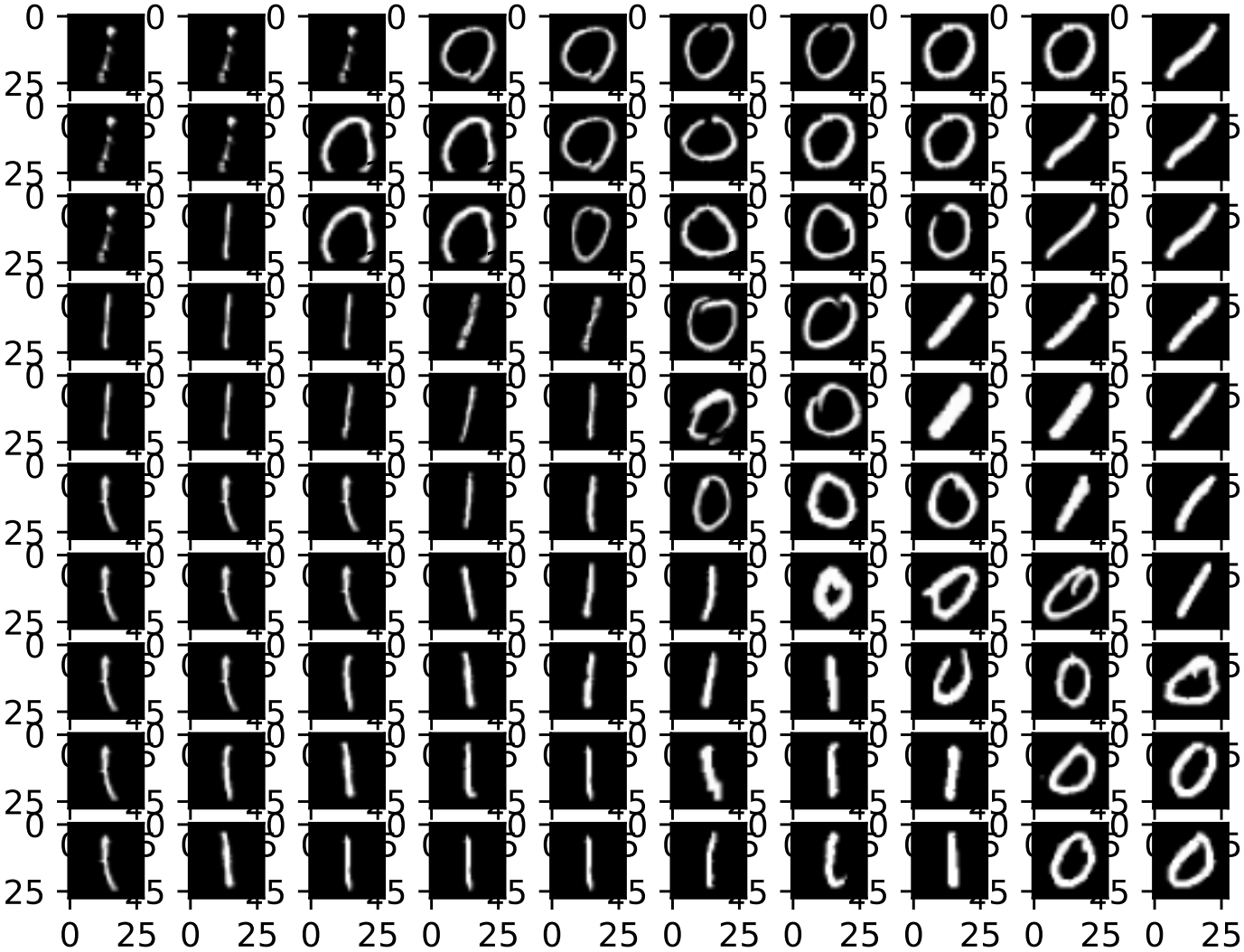

x
0
value
timestep
3
2
1
0
−1
0
500
1000
1500
2000
2500
3000
3500


x
0
1
2
value
timestep
3
2
1
0
−1
0
500
1000
1500
2000
2500
3000
3500

Interpretability

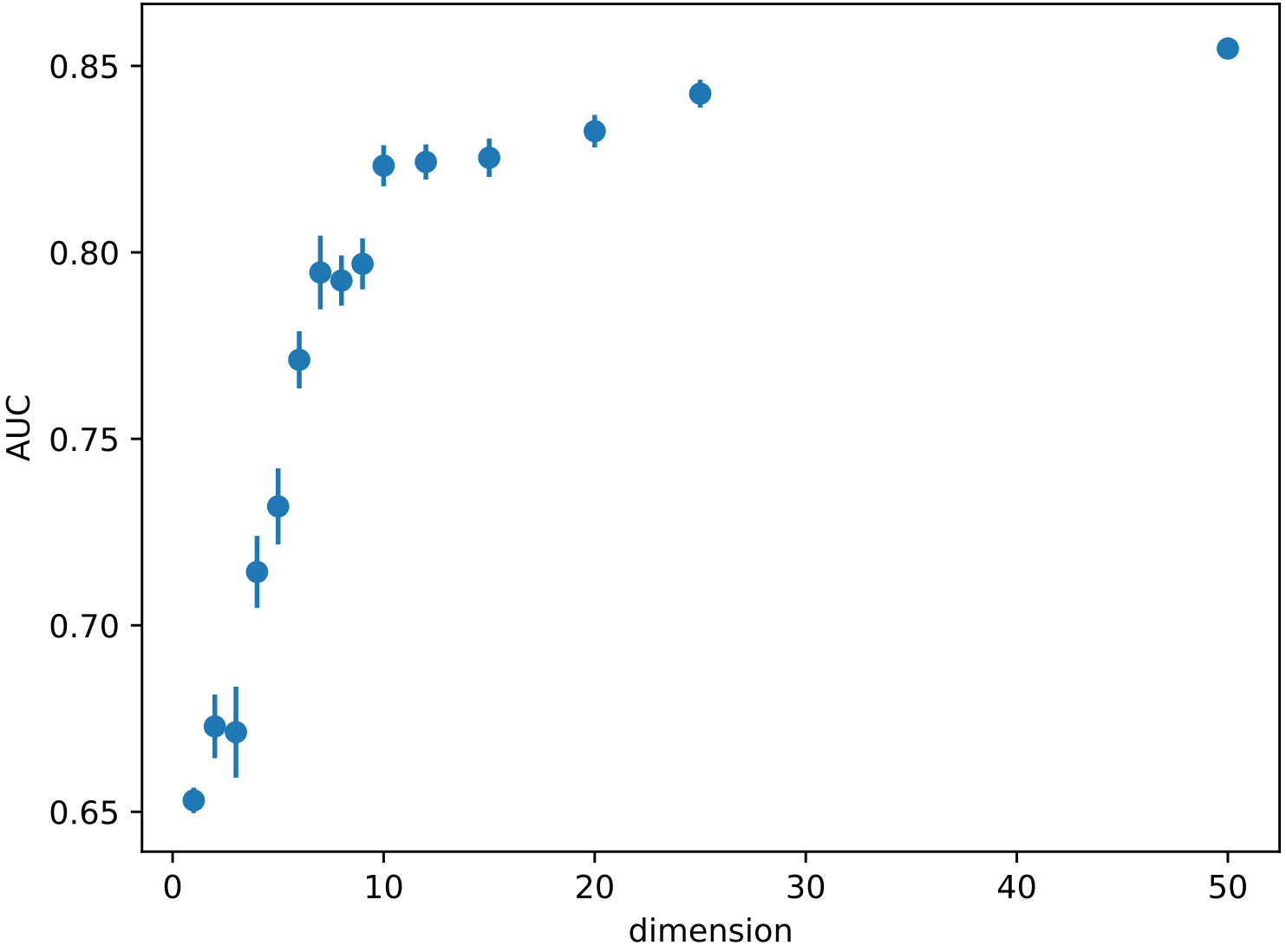
- By having an arbitrary number of output dimensions, you migth be able to find meaning in them
- Here looking at MNIST with 0 1 as normal case

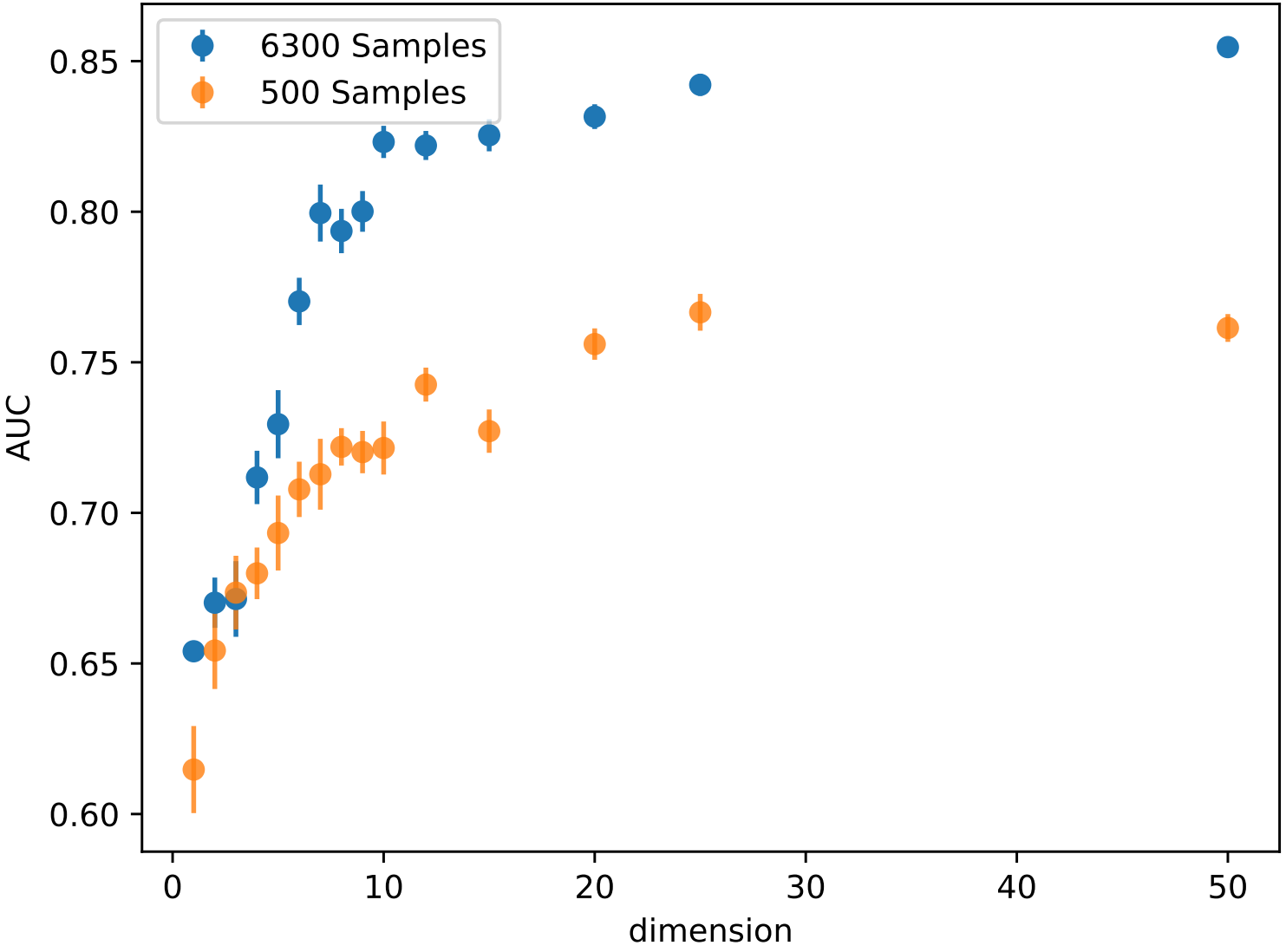




Strength

- Not ignoring (weak) features leads to better performance
- Also: this is able to combat the curse of dimensionality
- Here looking at one layer mnist anomaly detection (7 as normal class)
- generally really bad ($AUC = 0.65$)
- for comparison:2007.06963 reaches 0.982


0.85
0.80
0.75
0.70
0.65
AUC
0
10
20
30
40
50
dimension


6300 Samples
500 Samples
0.85
0.80
0.75
0.70
0.65
0.60
AUC
0
10
20
30
40
50
dimension

Whats next

- More efficient algorithm
- Try out other kinds of hard to find anomalies
- Better method of combining partial anomaly scores
- Other algorithms
  - Autoencoder, GAN based
  - classical ones (knn, isolation forest, svm)
- Try to prevent Feature mixing for interpretability
- Ability to train supervised algorithms on unsupervised output

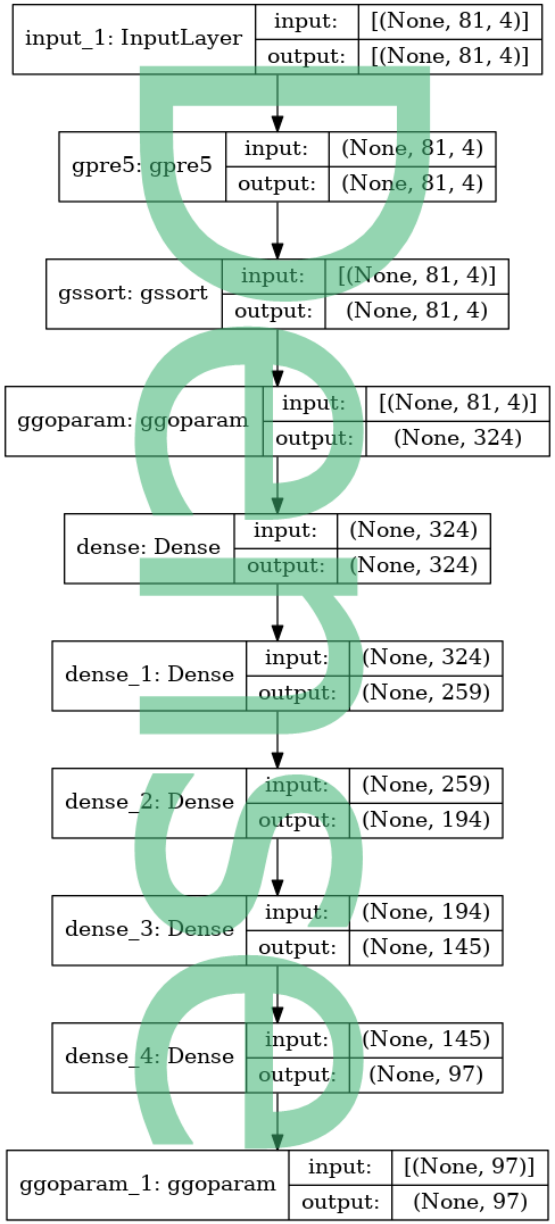
Thank You!

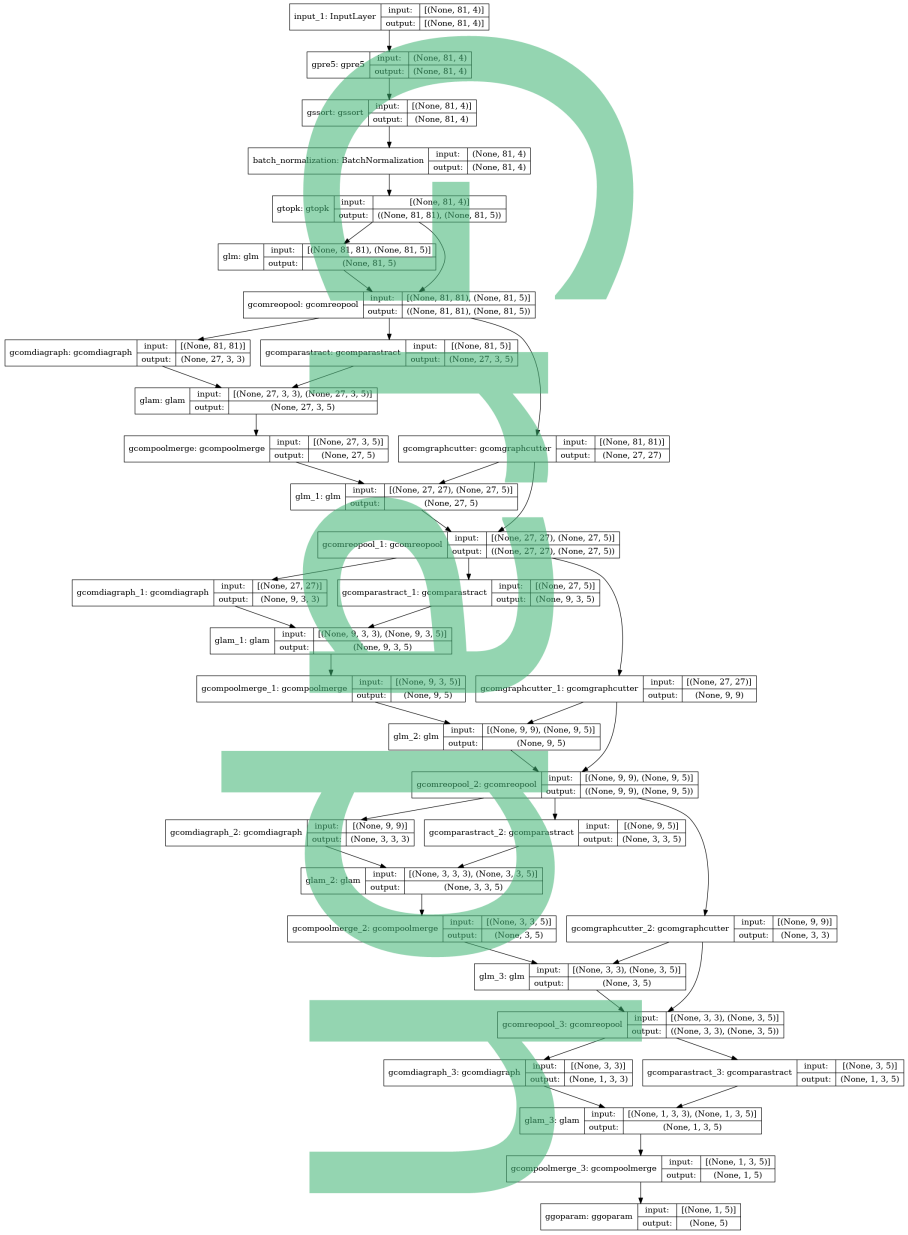
- So here ParticleNet + QCDorWhat
- =Graph Autoencoder
- To do this, require:
  - An update step
    - already described
  - something to reduce the number of nodes
  - something to increase the number of nodes afterwards again

Going from a big graph to a small graph

- 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)

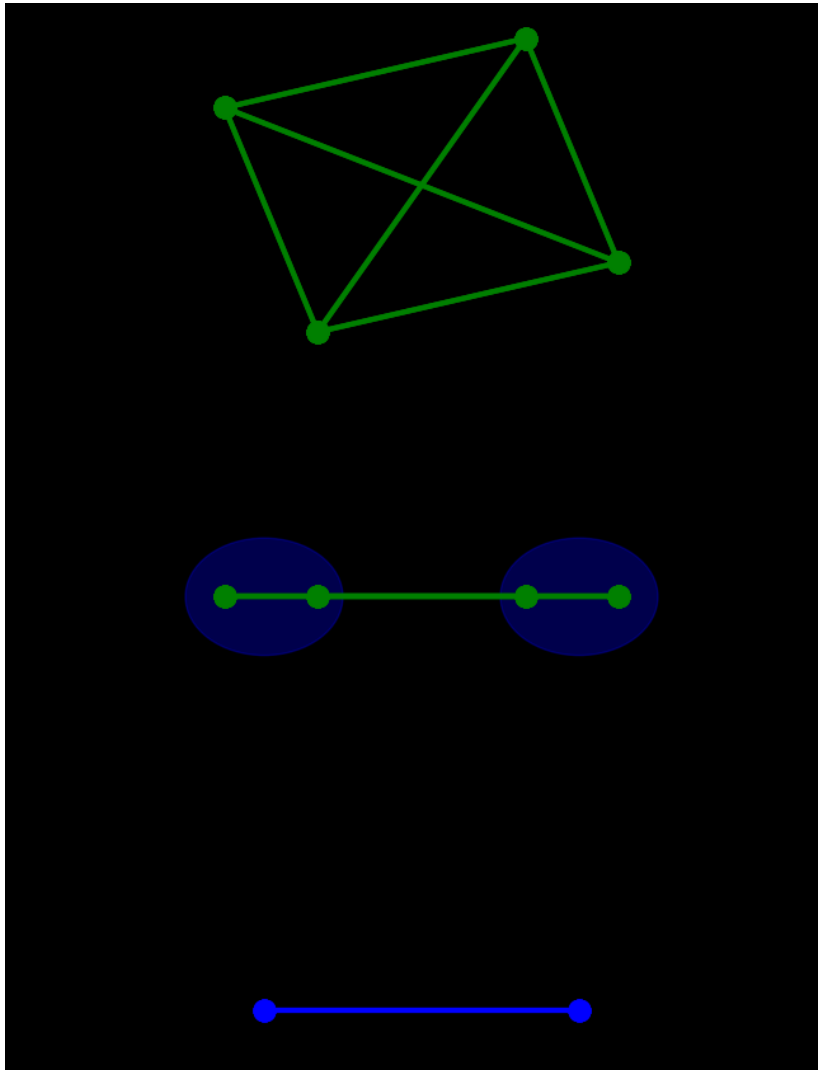
Backup Graph Autoencoder

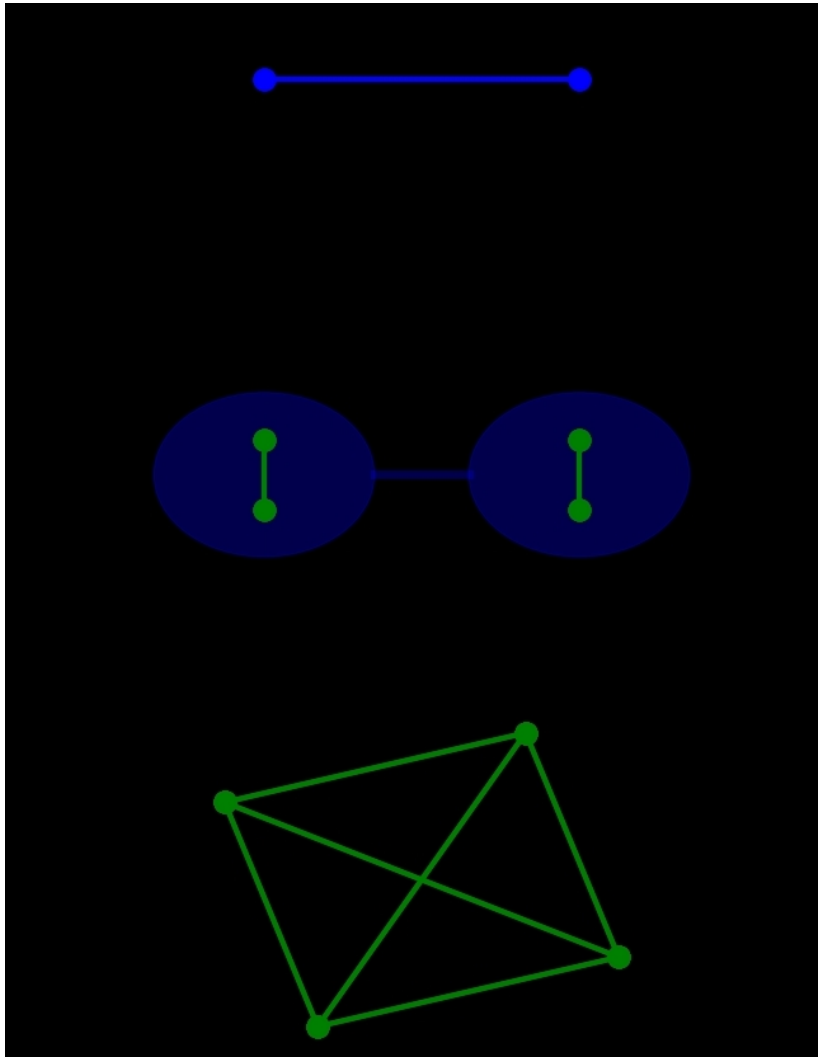




from a big graph to a small graph

- project the graph on a learnable axis
- combine neigbour nodes on this axis
- relearn the graph or use a graph combination rule



from a small graph to a big graph

- let each node grow into a learnable graph
- combine the new graphs with the existing one