IX. Jet substructure with Convolutional Neural Networks

1) Jets and jet substructure

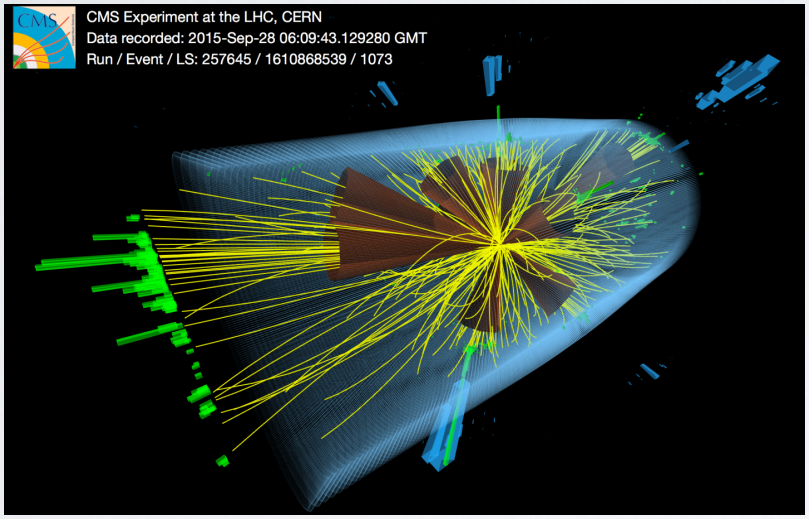
2) Quick start guide to CNNs

3) Example: searching for boosted Higgs bosons inside jets

1)

Jets and Jet Substructure

A **jet** is a collimated collection of hadronic activity in an event



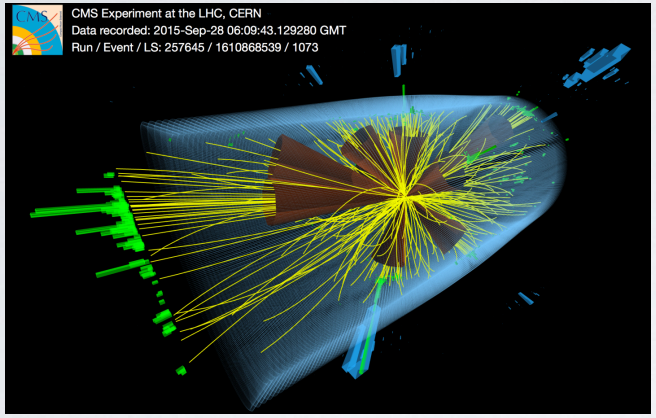
Feature of **strongly interacting** radiation which we model with **QCD**

Can be identified by **clustering** hadronic momenta according to different **distance measures** ⇒ different types of jets and Jet algorithms

1)

Jets and Jet Substructure

clustering algorithms



at hadron colliders we must also require a minmum p_T to ensure no initial state collinear divergence

$$d_{iB} = p_T^{2A}$$

cluster partons/hadrons if the distance measure with respect to a jet radius R, e.g.

$$d_{ij} = \min(p_{T,i}^{2A}, p_{T,j}^{2A}) \frac{\Delta R_{ij}^2}{R}$$

$$\Delta R_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$$

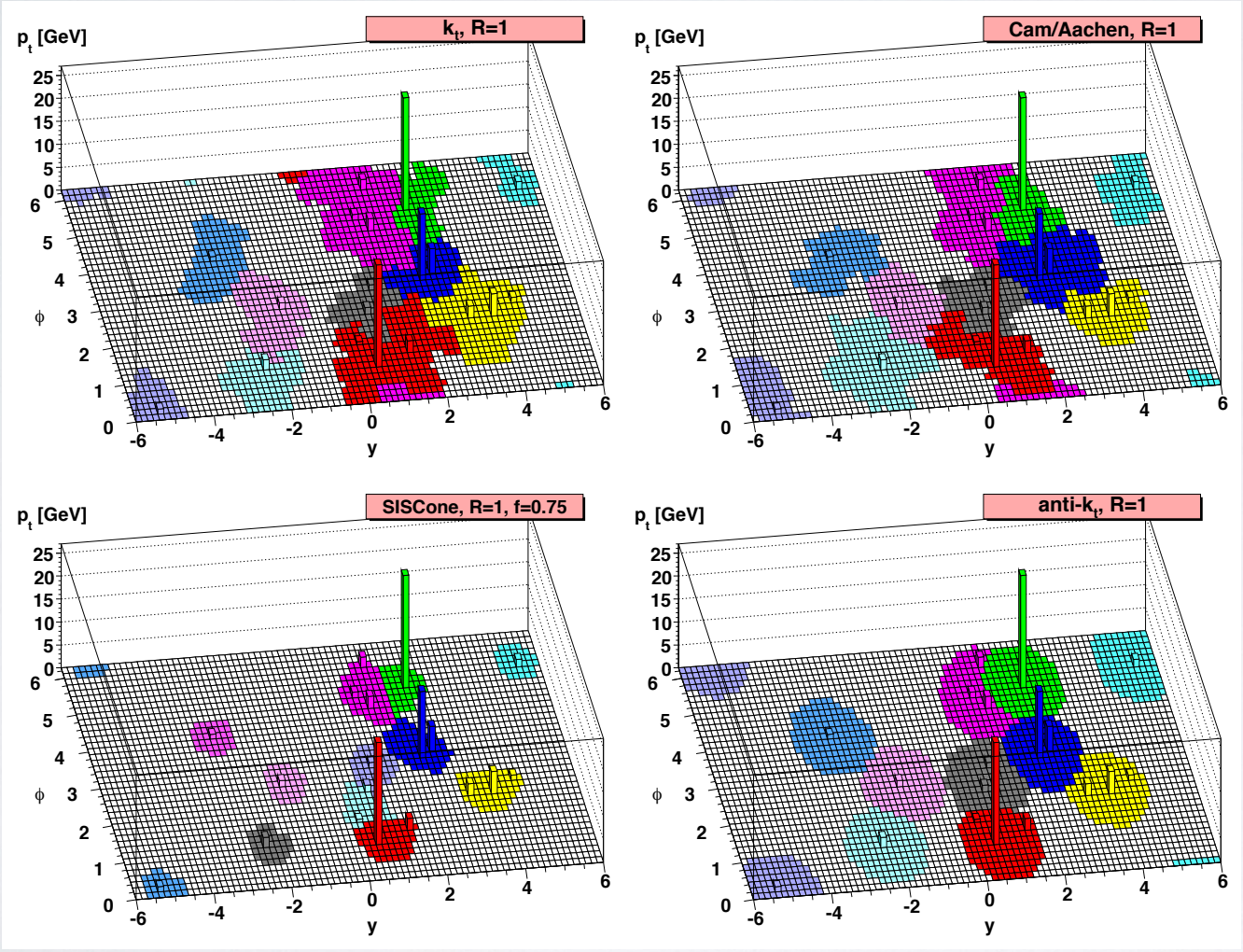
A=1 'k_T algorithm'

A=0 'Cambridge/Aachen algorithm'

A=-1 'anti-k_T algorithm'

1)

Jets and Jet Substructure



clustering with different distance measures leads do different shapes

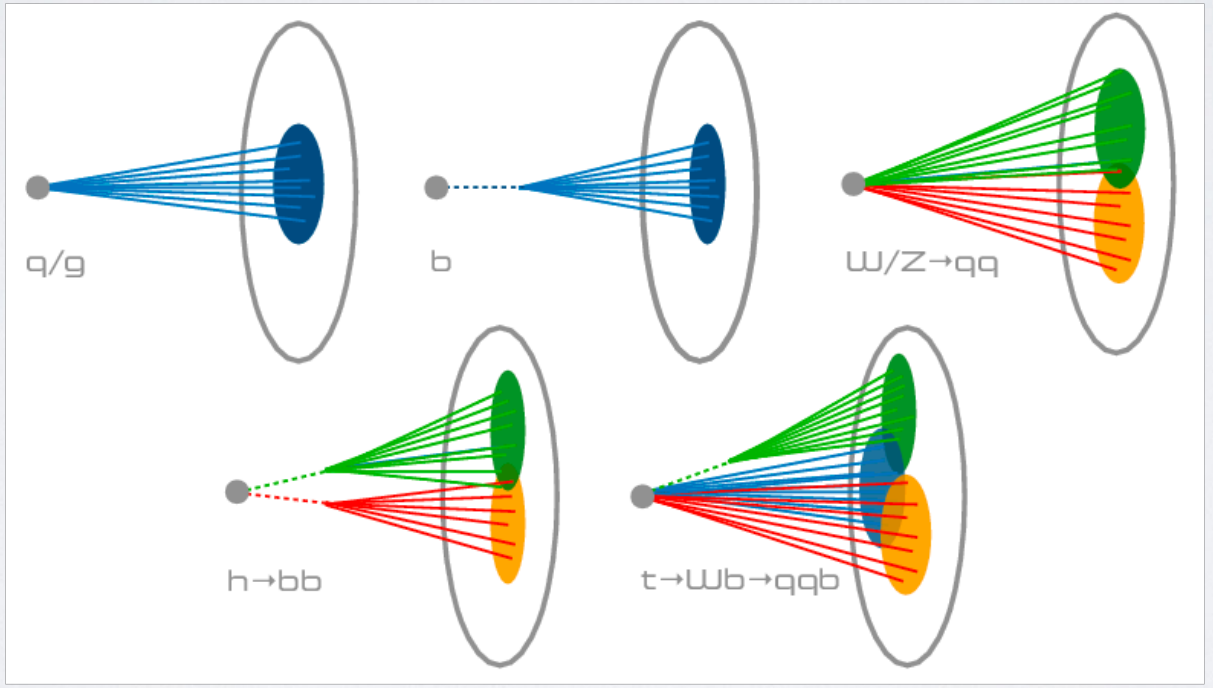
there are also cone algorithms but a more difficult to define in an infrared safe manner

1)

Jets and Jet Substructure

Having identified regions of hadronic activity we may ask how the momenta that make up the jet are distributed

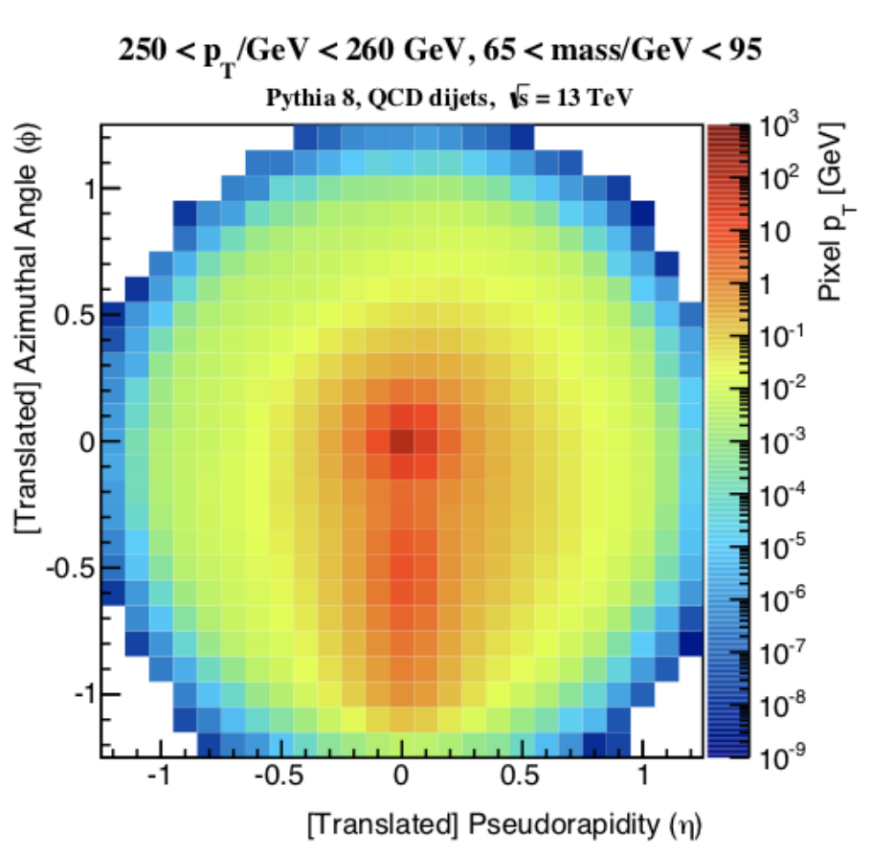
This structure can tell us something about the sequence of emissions that formed the jet



1)

Jets and Jet Substructure

After the jets in an event have been identified we may represent them as **jet images** in the η-φ plane



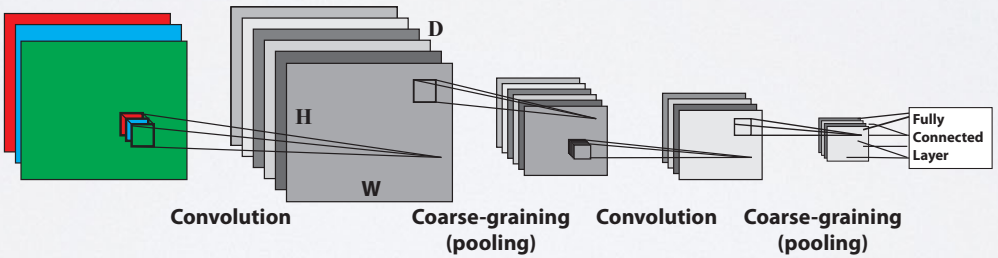
2)

Quick start guide to CNNs

many online resources, e.g. http://cs231n.stanford.edu/

Specialsed networks designed for image processing

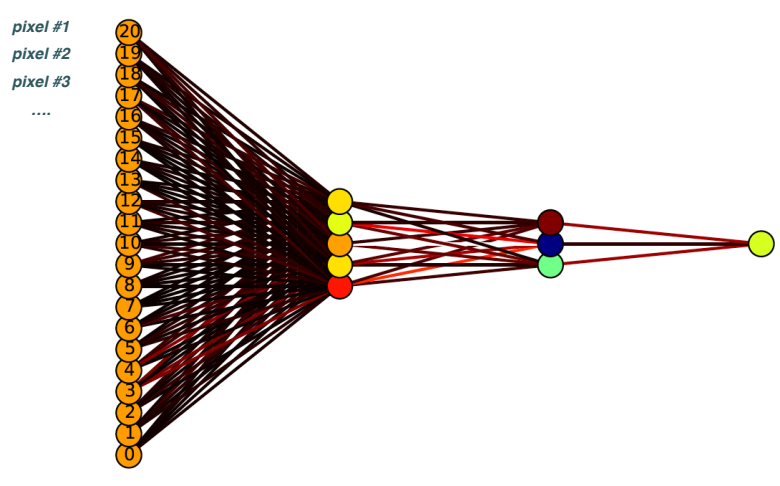
Important to exploit translational and rotational symmetry
to reduce the training to a manageable problem



the full network consists of different types of layers
which perform specific operations

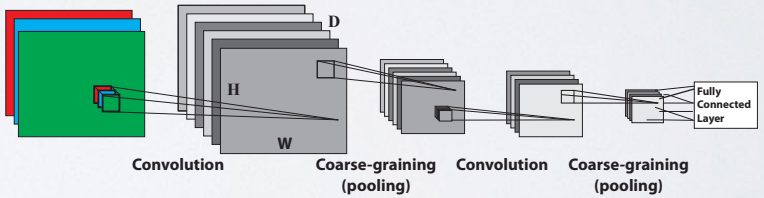
2)

Quick start guide to CNNs



We could try to classify images with a large fully connected DNN - it would have to learn the symmetries and would have huge parameter space

nearby pixels will be **correlated** so we first attempt to identify **local** features



2)

Quick start guide to CNNs

Our image data will be in the form of a n x n 2d pixel grid each with a colour value:

(width x height) x depth

= (pixel grid) x (colours)

Types of layers:

Dense: exactly as we have been using in our regression examples for a fully connected neural network - layer ***depth***, d, of neurons

Convolutional: 3d grid of neurons with ***depth***, ***width*** and ***height***. The layer is designed to identify low level features such as edges.

Pooling: Apply coarse graining to the input image, reducing the grid size while maintaining the spatial structure

Batch Normalisation: Often added to re-normalise the standard deviation and mean as usually done with input layer. Optimises fitting

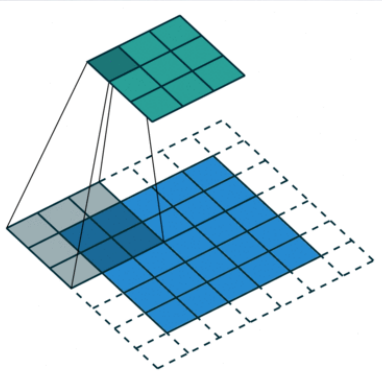
2)

Quick start guide to CNNs

Convolutional: 3d grid of neurons with ***depth***, ***width*** and ***height***. The layer is designed to identify low level features such as edges.

The idea is to **scan the image** by processing smaller **subregions**. The weights and biases of a standard fully connected neuron are replaced with **filters** which determine how each subregion is processed.

Scanning subregions with the same set of weights and biases gives the convolutional network **translation invariance**



The convolutional layer is determined by the hyperparameters: number of filters (F), the spatial extent of the filters (S), the stride (T), and the amount of zero padding (P)

input image:

$W_1 \times H_1 \times D_1$

pixels

colours

output image:

$W_2 \times H_2 \times D_2$

$$D_2 = F$$

$$W_2 = (W_1\text{-}S+2P)/T+1$$

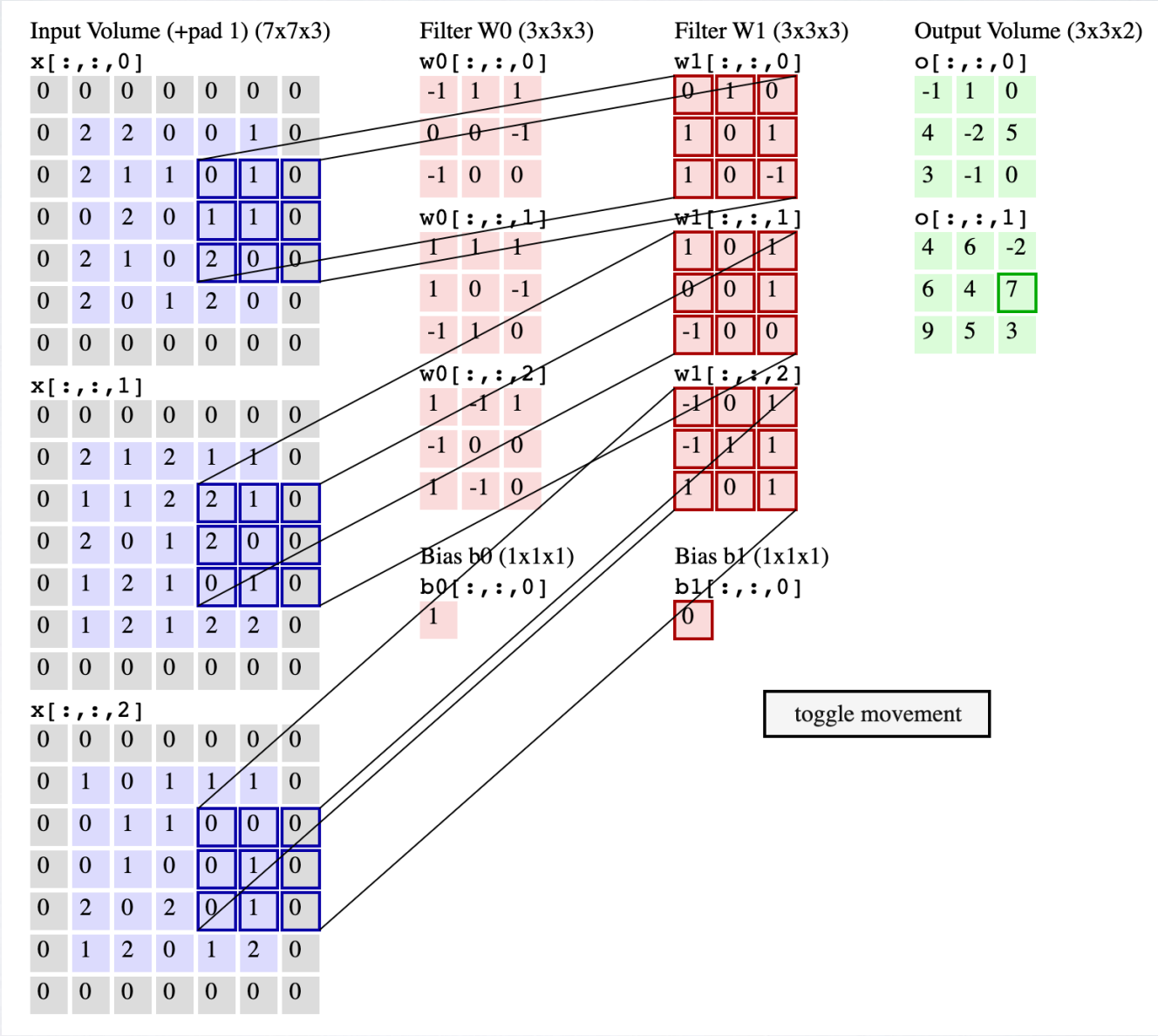
$$H_2 = (H_1\text{-}S+2P)/T+1$$

2)

Quick start guide to CNNs

https://cs231n.github.io/convolutional-networks/

visual example:
follow the link to
watch the scan



2)

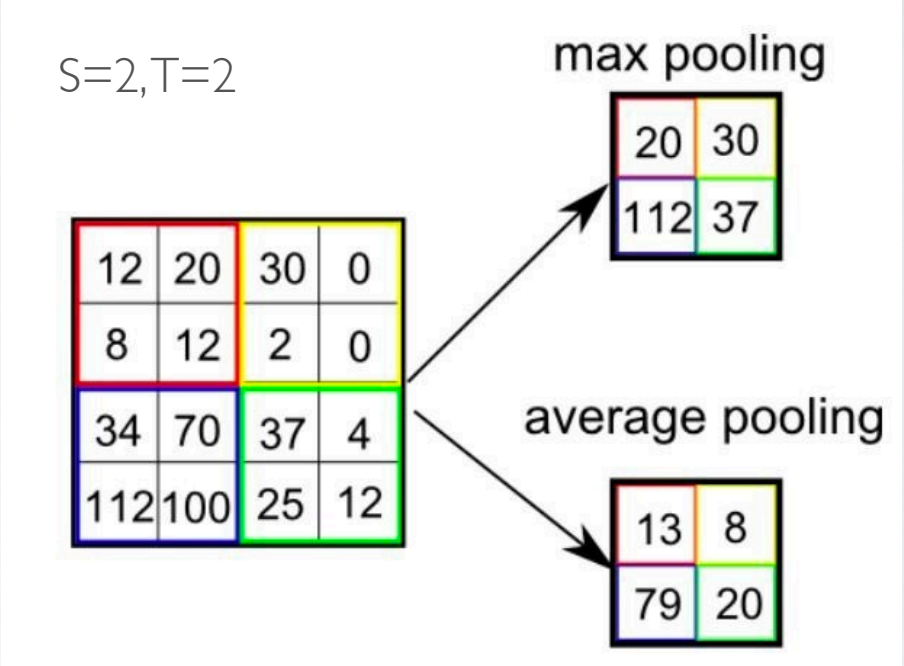
Quick start guide to CNNs

Pooling: Apply coarse graining to the input image, reducing the grid size while maintaining the spatial structure

The pooling layer is determined by the hyperparameters: the spatial extent (S) and the stride (T)

Subregions of size S are combined to a single value be averaging to taking maximum value

Image is split into subregions according to the stride T



input image:
$W_1 \times H_1 \times D_1$

output image:
$W_2 \times H_2 \times D_2$

$D_2 = D_1$

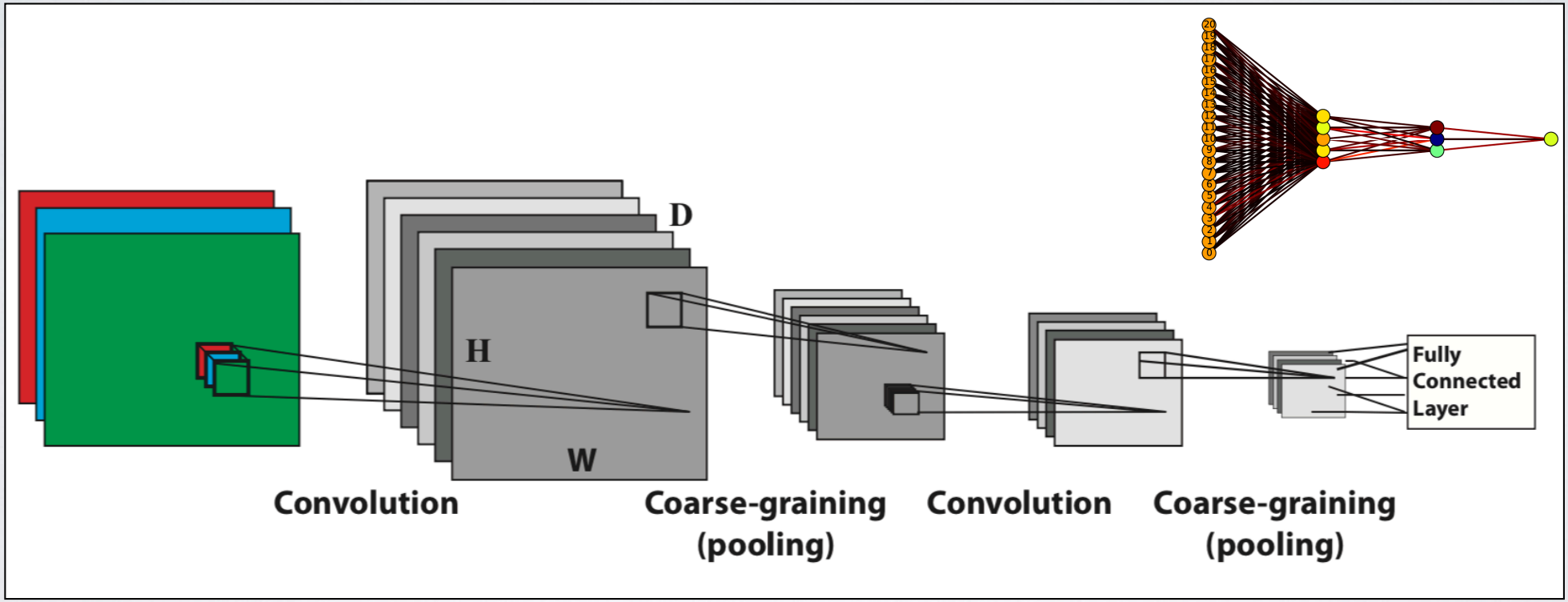
$W_2 = (W_1\text{-}S)/T+1$

$H_2 = (H_1\text{-}S)/T+1$

2)

Quick start guide to CNNs

After processing with convolutional and pooling layers the image is flattened and passed to a fully connected network

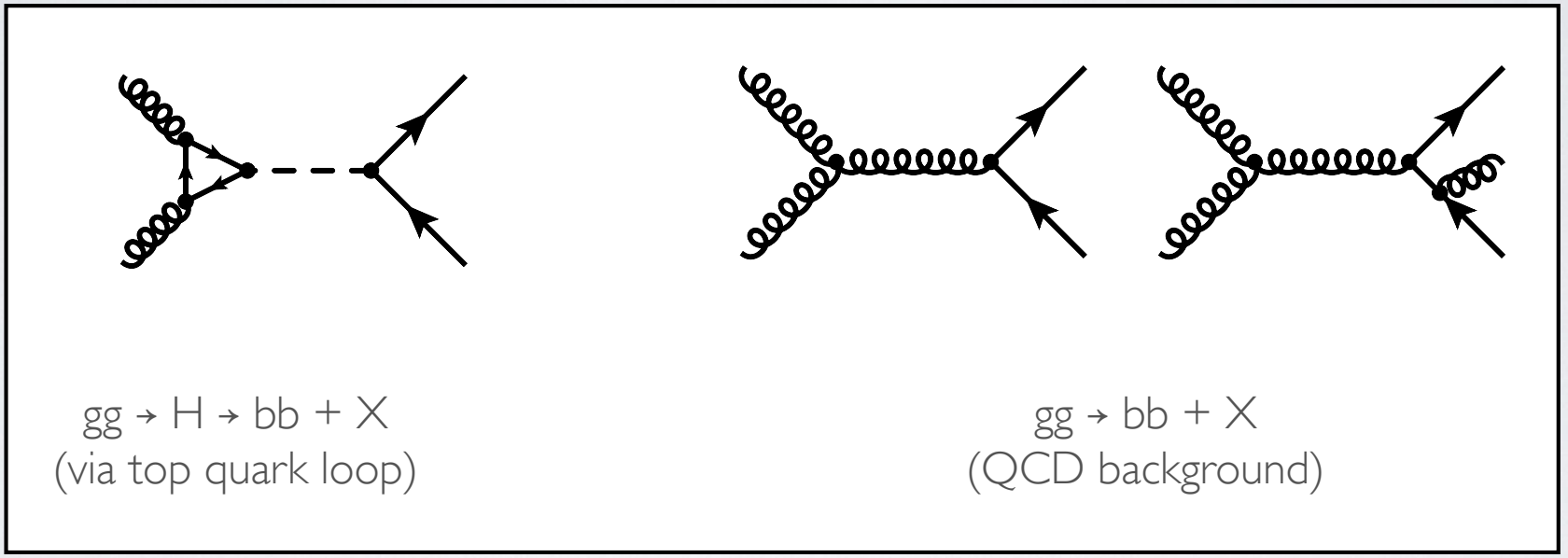


All sorts of image classification examples online - go check them out!

3)

Example: searching for boosted Higgs bosons inside jets

While the structure of the CNN is quite complicated, it is actually quite easy to implement within the Keras/Tensorflow framework. Let's see how it works on a physics example.



gg → H → bb + X
(via top quark loop)

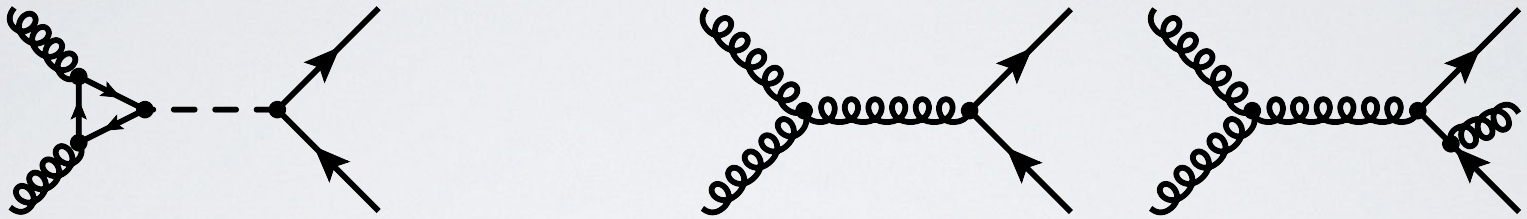
gg → bb + X
(QCD background)

example taken from

http://jduarte.physics.ucsd.edu/capstone-particle-physics-domain/weeks/05-jet-images.html

3.1)

Event generation

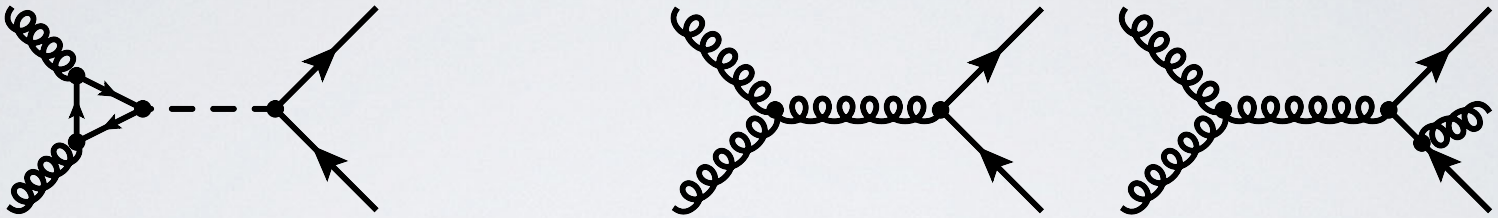


The first step is to **simulate events** with and MC event generator which may then run through a jet algorithm and extract jet image data for training

We would like to account for the strong radiation pattern observed in the hadron level measurements seen in the detector and so include the effects of the **parton shower**

3.1)

Event generation



The simplest thing to do is simulate using **Leading Order** (LO) amplitudes matched to the **Parton Shower** (PS)

This can be achieved with multi-purpose generators such as **Madgraph** (for LO) and **Pythia** (for PS)

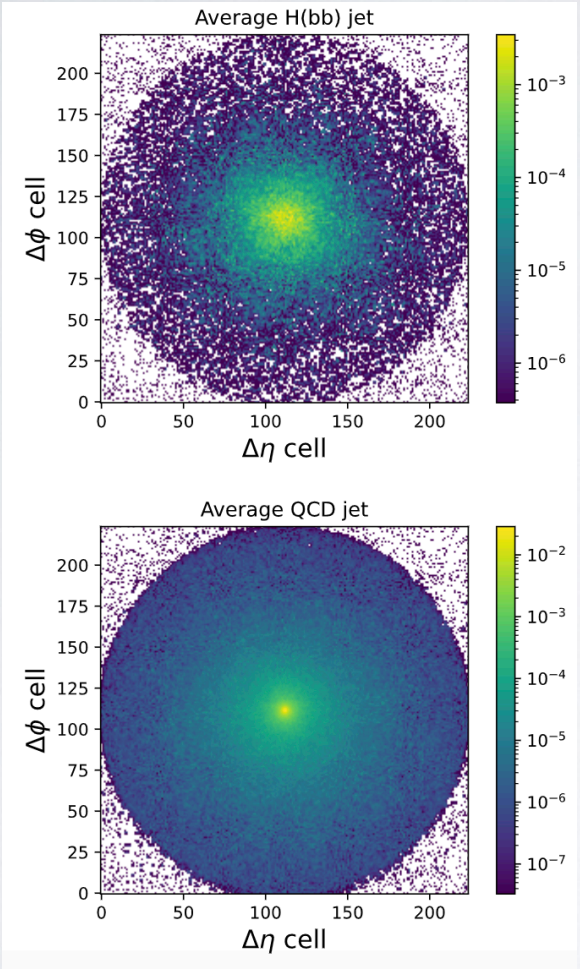
A set of events generated by the CMS collaboration (along with a lot of other things!) has been prepared as part of the CMS open data project

http://opendata.cern.ch/docs/about-cms

3.2)

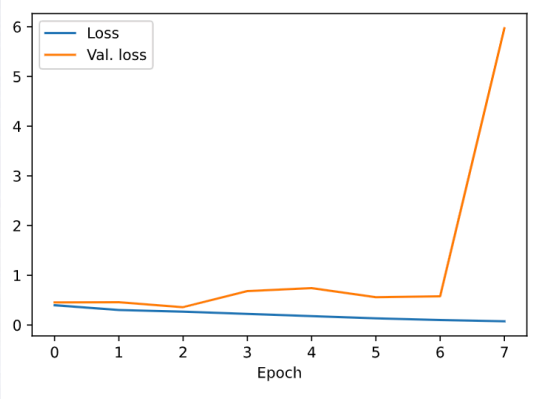
Example notebook

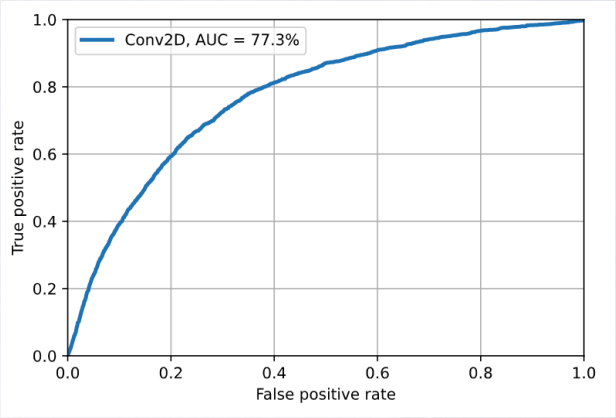
`CNN_jetclassification.ipynb`



demonstration of how to set up CNN in tensorflow and train a simple classifier

```
=================================
Total params: 6,452,166
Trainable params: 6,451,908
Non-trainable params: 258
```



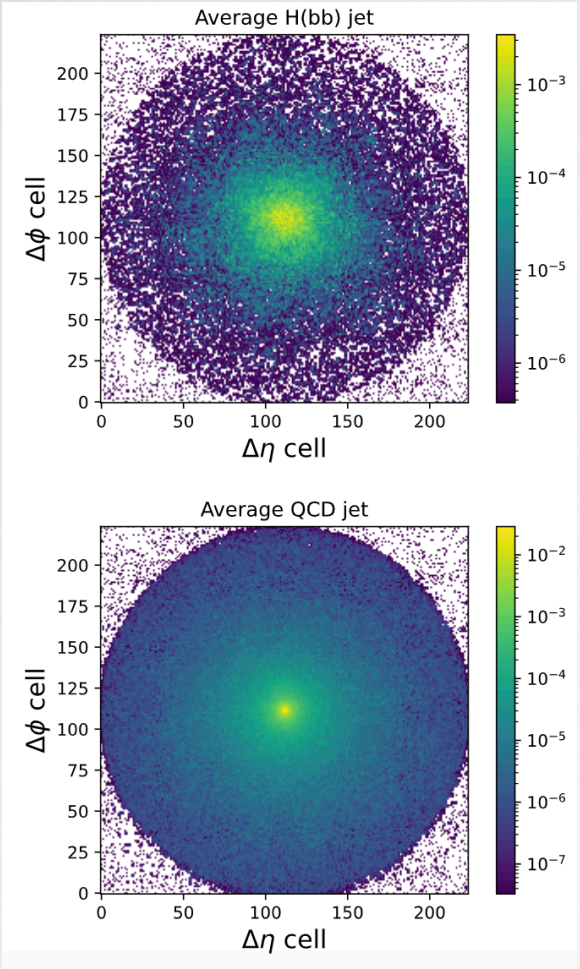


training is quite CPU intensive due to large parameter space. Optimisations on GPUs are often used in these cases.

3.2)

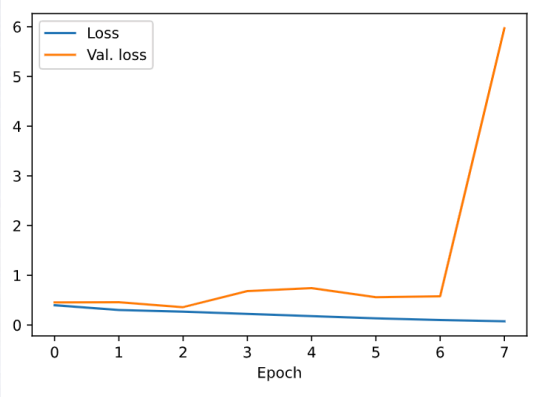
Example notebook

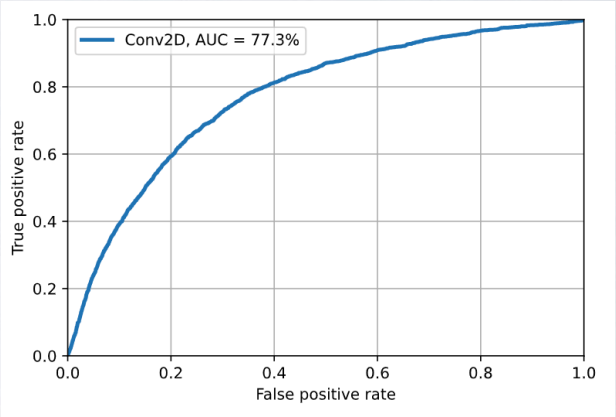
`CNN_jetclassification.ipynb`



demonstration of how to set up CNN in tensorflow and train a simple classifier

```
=================================
Total params: 6,452,166
Trainable params: 6,451,908
Non-trainable params: 258
```





training is quite CPU intensive due to large parameter space. Optimisations on GPUs are often used in these cases.

remark: how can we generalise the classifier to generate new jet images? leads us to **generative models**