Machine learning Project

Topic: Application of convoultional neural nets to identify jets at LHC

Shivam Verma
Research scholar, Dept. of physics
RKMVERI

Dataset Visualization

File Content

In [2]:
```
ip = 'Data/JetDataset/jetImage_7_100p_30000_40000.h5'
f = h5py.File(ip)
print(list(f.keys()))
```

```
['jetConstituentList', 'jetFeatureNames', 'jetImage', 'jetImageECAL', 'jetImageHCAL', 'jets', 'particleFeatureName
s']
```

- 'jetImage' contains the image representation of the jets (vector representing each jet).
- 'jetImageECAL' and 'jetImageHCAL' are the ECAL- and HCAL-only equivalent images. not being used.
- 'jetConstituentList' is the list of particles cointained in the jet. For each particle, a list of relevant quantities is stored
- 'particleFeatureNames' is the list of the names corresponding to the quantities contained in 'jetConstituentList'
- 'jets' is the dataset to work with.
- 'jetFeatureNames' is the list of the names corresponding to the quantities contained in 'jets'

Jets data

```
Dataset shape: (10000, 5)
First five entries:
[1. 0. 0. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 0. 0. 0. 1.]
[1. 0. 0. 0. 0.]
[0. 0. 0. 1. 0.]
Last 5 entries:
[0. 0. 1. 0. 0.]
[1. 0. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[0. 1. 0. 0. 0.]
[1. 0. 0. 0. 0.]
```

- [1, 0, 0, 0, 0] for gluons
- [0, 1, 0, 0, 0] for quarks
- [0, 0, 1, 0, 0] for Ws
- [0, 0, 0, 1, 0] for Zs
- [0, 0, 0, 0, 1] for Tops

Each of them are represent a unique particle with unique signature

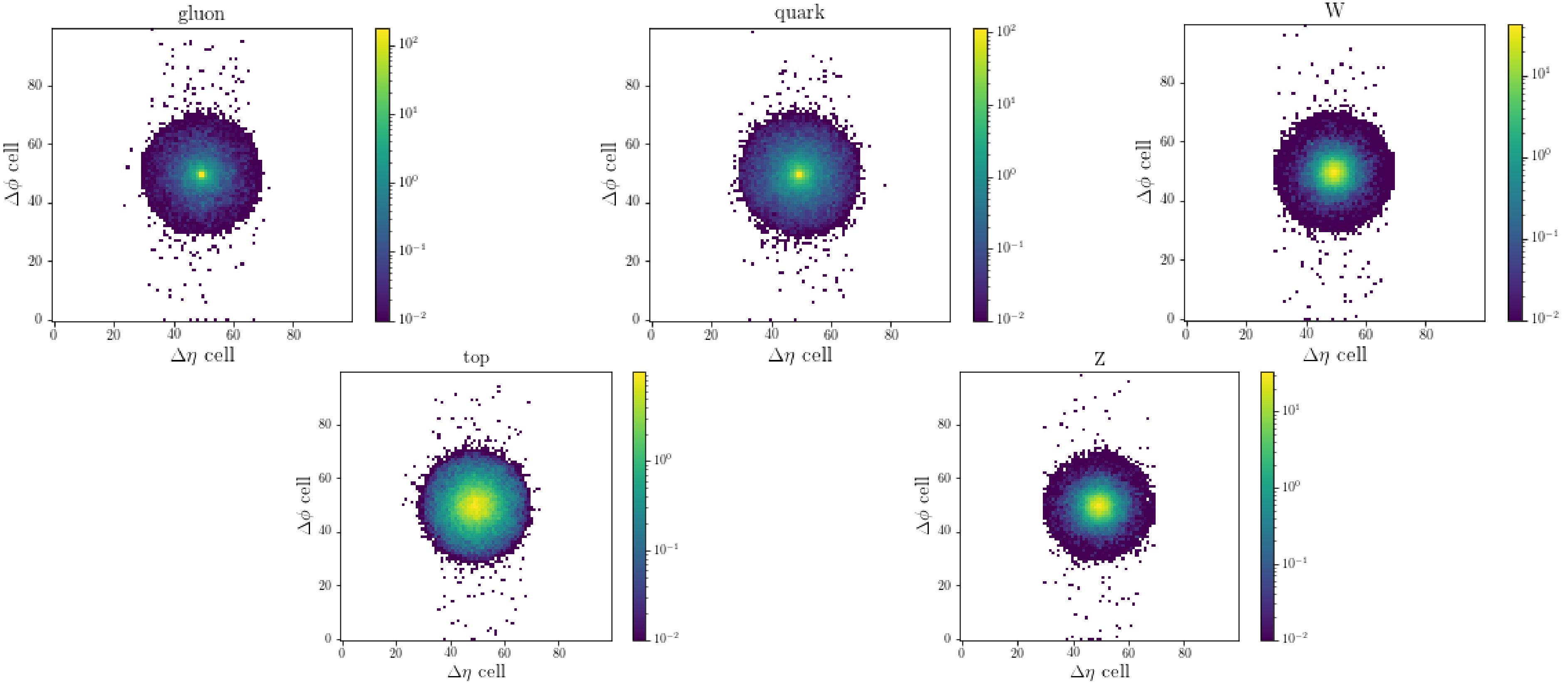
Features

In [10]:
```
featurenames = f.get('jetFeatureNames')
print(featurenames[:])
```

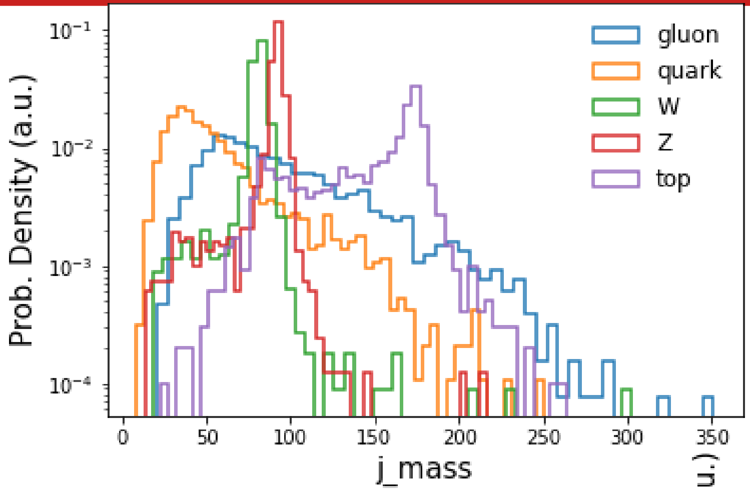
```
[b'j_ptfrac' b'j_pt' b'j_eta' b'j_mass' b'j_tau1_b1' b'j_tau2_b1'
 b'j_tau3_b1' b'j_tau1_b2' b'j_tau2_b2' b'j_tau3_b2' b'j_tau32_b1'
 b'j_tau32_b2' b'j_zlogz' b'j_c1_b0' b'j_c1_b1' b'j_c1_b2' b'j_c2_b1'
 b'j_c2_b2' b'j_d2_b1' b'j_d2_b2' b'j_d2_a1_b1' b'j_d2_a1_b2' b'j_m2_b1'
 b'j_m2_b2' b'j_n2_b1' b'j_n2_b2' b'j_tau1_b1_mmdt' b'j_tau2_b1_mmdt'
 b'j_tau3_b1_mmdt' b'j_tau1_b2_mmdt' b'j_tau2_b2_mmdt' b'j_tau3_b2_mmdt'
 b'j_tau32_b1_mmdt' b'j_tau32_b2_mmdt' b'j_c1_b0_mmdt' b'j_c1_b1_mmdt'
 b'j_c1_b2_mmdt' b'j_c2_b1_mmdt' b'j_c2_b2_mmdt' b'j_d2_b1_mmdt'
 b'j_d2_b2_mmdt' b'j_d2_a1_b1_mmdt' b'j_d2_a1_b2_mmdt' b'j_m2_b1_mmdt'
 b'j_m2_b2_mmdt' b'j_n2_b1_mmdt' b'j_n2_b2_mmdt' b'j_mass_trim'
 b'j_mass_mmdt' b'j_mass_prun' b'j_mass_sdb2' b'j_mass_sdm1'
 b'j_multiplicity' b'j_g' b'j_q' b'j_w' b'j_z' b'j_t' b'j_undef']
```

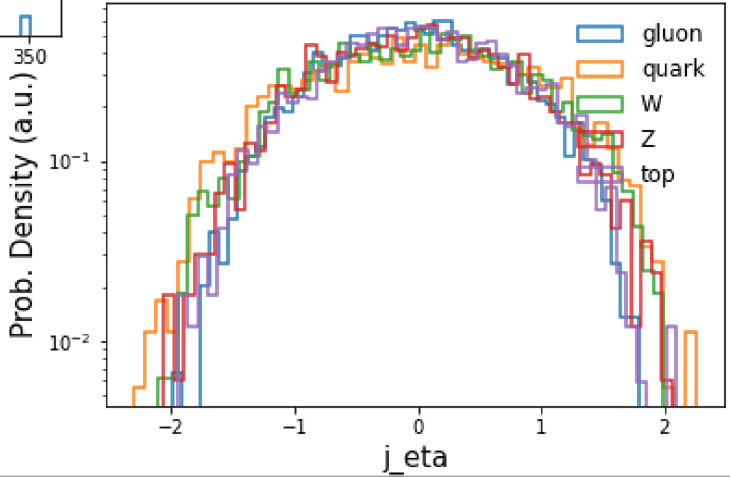
A total of 53 features

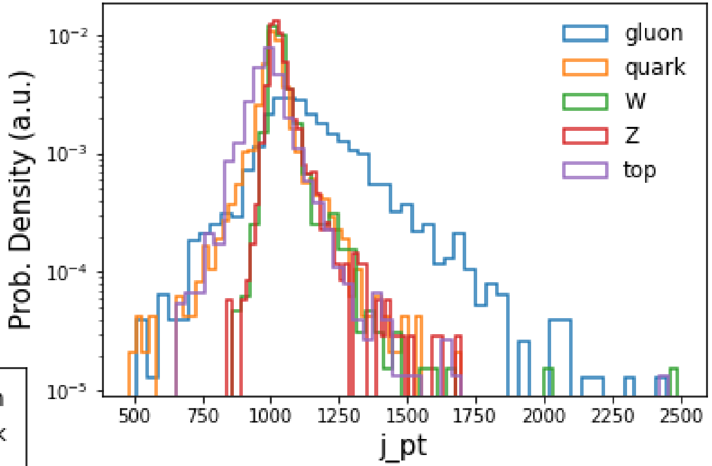
Image of particle jets in $\Delta\phi$ Vs $\Delta\eta$ plane



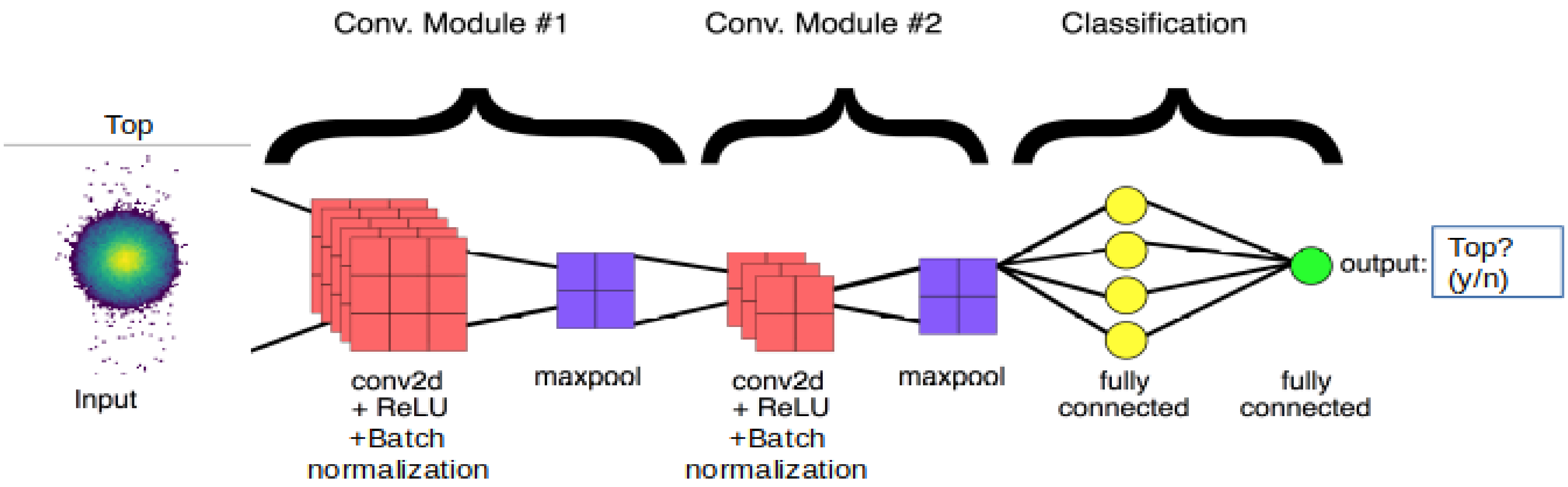
A few feature plots







Methodology for analysis



Convolutional filter

- The main ingredient of Convolutional neural network is a filter, i.e. a n x n' matrix to scan acoss the image matrix.
- The filter scans the image and performs a scalar product of each image patch while sliding through the whole image.
- This results into a new matrix of values, with different dimensionality.

0	3	5	6	2	4	5
7	4	7	3	6	3	4
9	1	2	1	9	6	0
9	2	1	1	7	3	5
8	0	4	7	6	8	0
8	3	4	5	5	3	4
7	9	4	6	5	2	6

4	-1	4
-2	2	-5
3	1	-6

3x4 - 5x1 + 6x4 +
-4x2 + 7x2 - 3x5 +
1x3 + 2x1 - 1x6 = 21

-8	21			

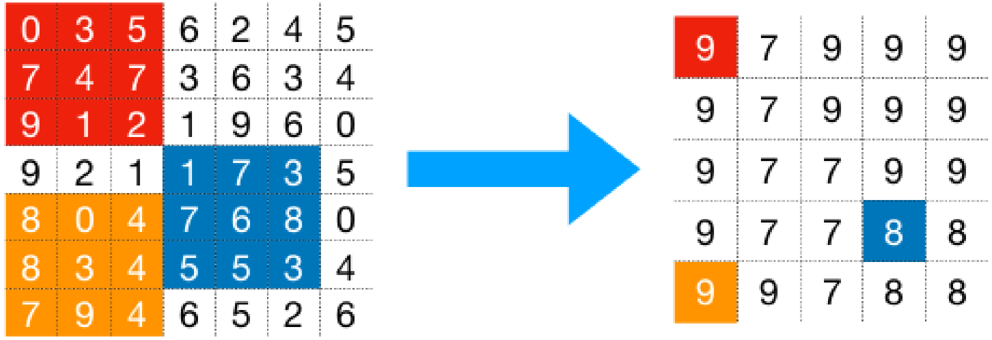
Source: GGI Lectures on ML by Maurizio pierini

Max pooling and padding

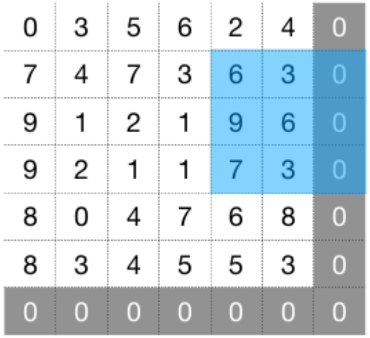
Max pooling: For a given image of m x m’ and a filter of size n x n’, scans the image and replaces each n x n’ patch with its maximum.

Padding:

- When the filter arrived at the edge, it might exceeds it (if m/n is not an integer).
- Padding can be of 2 types: **same**(the last entry is repeated) and **zero** (padded with 0 as the entry)



Same padding



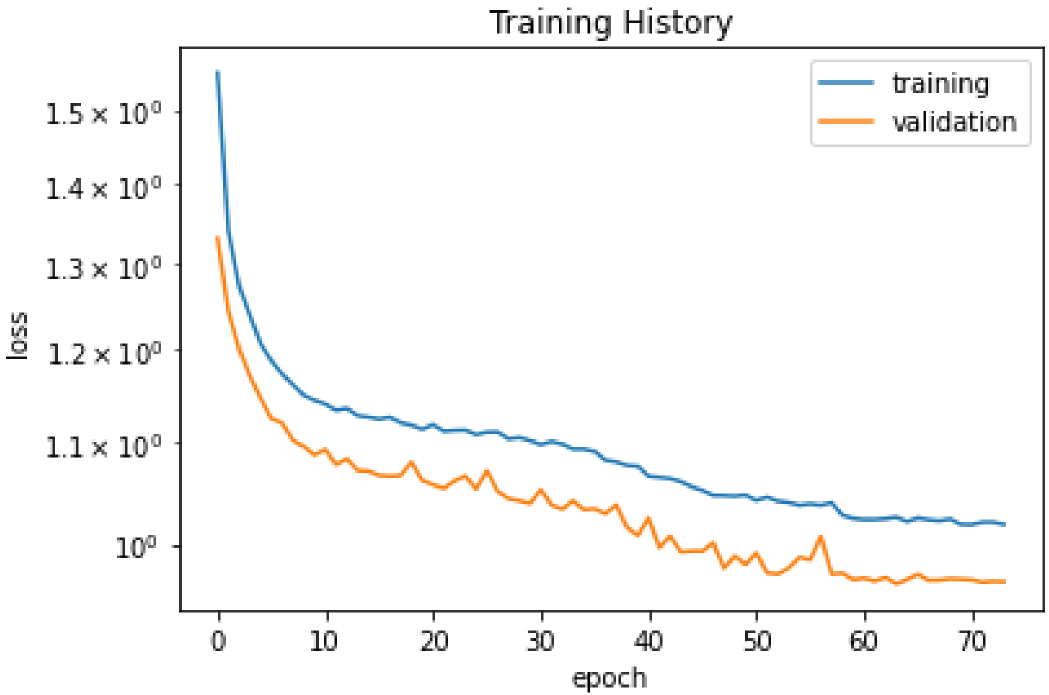
Zero padding

Model Summary

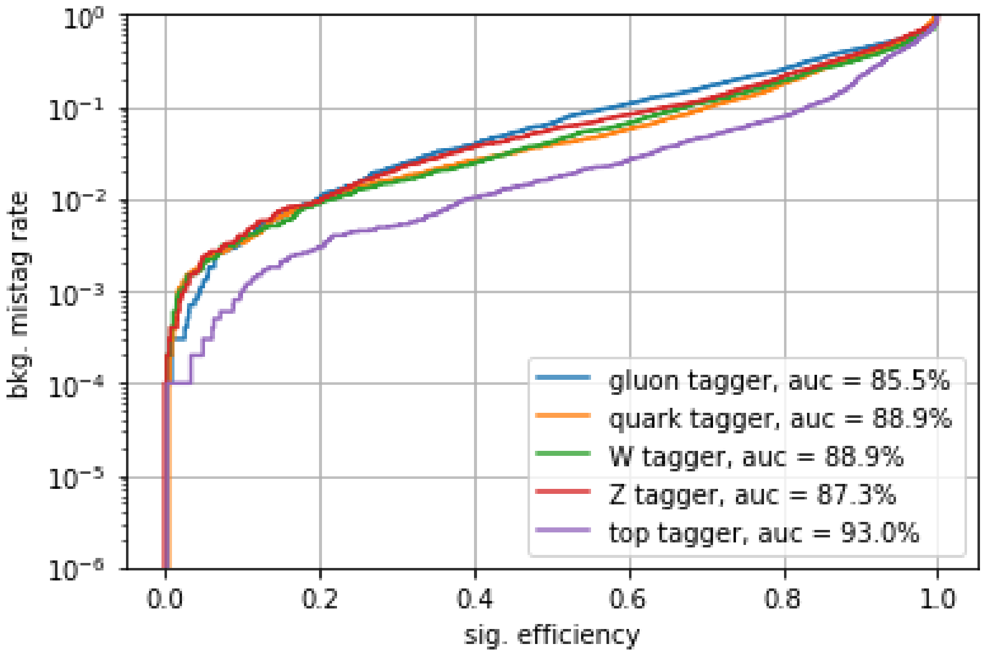
```
model.compile(loss='categorical_crossentropy', optimizer='adam')
model.summary()
```

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 100, 100, 1)]     0
_________________________________________________________________
conv2d (Conv2D)              (None, 100, 100, 5)       130
_________________________________________________________________
batch_normalization (BatchNo (None, 100, 100, 5)       20
_________________________________________________________________
activation (Activation)      (None, 100, 100, 5)       0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 20, 20, 5)         0
_________________________________________________________________
dropout (Dropout)            (None, 20, 20, 5)         0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 20, 20, 3)         138
_________________________________________________________________
batch_normalization_1 (Batch (None, 20, 20, 3)         12
_________________________________________________________________
activation_1 (Activation)    (None, 20, 20, 3)         0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 3)           0
_________________________________________________________________
dropout_1 (Dropout)          (None, 6, 6, 3)           0
_________________________________________________________________
flatten (Flatten)            (None, 108)               0
_________________________________________________________________
dense (Dense)                (None, 5)                 545
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 30
=================================================================
Total params: 875
Trainable params: 859
Non-trainable params: 16
```

Results



Training History
training
validation
loss
epoch
1.5×10^0
1.4×10^0
1.3×10^0
1.2×10^0
1.1×10^0
10^0
0
10
20
30
40
50
60
70



bkg. mistag rate
sig. efficiency
gluon tagger, auc = 85.5%
quark tagger, auc = 88.9%
W tagger, auc = 88.9%
Z tagger, auc = 87.3%
top tagger, auc = 93.0%
10^0
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
0.0
0.2
0.4
0.6
0.8
1.0