Optical Processing Units In HEP

(Preliminary Research: Event Classification)

Biswajit Biswas

Learning to Discover: Advanced Pattern Recognition, IPa Workshop
17th October, 2019

Biswajit Biswas, Aishik Ghosh, David Rousseau (LAL-Orsay) Laurent Basara (LRI-Orsay)

Many thanks to the LightOn team in particular Laurent Daudet, Iacopo Poli
and
Steve Farrell, Wahid Bhimji for the dataset and useful discussions

Event Classification

Detecting signals which has some characteristic traits can be represented as a problem of event classification of **signal against background**.

Some of these problems can be treated as an **image classification** problem by treating the data from calorimeter as images where pixel values represent the intensity of energy in corresponding regions of the calorimeter.

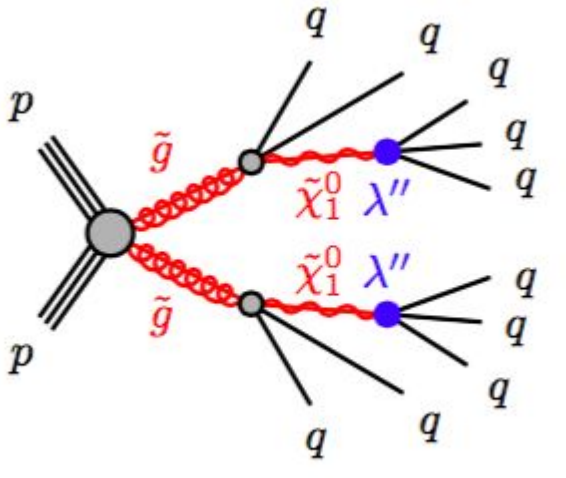
Data

Search for RPV SUSY gluino decays

- Multi-jet final state
- Analysis from ATLAS-CONF-2016-057 used as a benchmark
- Classification problem: RPV Susy vs. QCD

Simulated samples

Pythia - event gen. (matching ATLAS config)
Cascade $m_{\tilde{g}} = 1400$, $m_{\tilde{\chi}0}=850$ default
Delphes detector simulation (ATLAS card)
- Output calorimeter towers (and tracks) used in analysis



gluino cascade decay

Representation of data as an Image

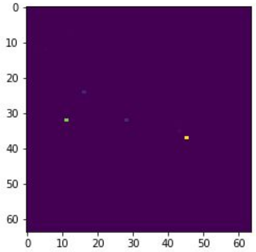


Fig 2. (a) Background

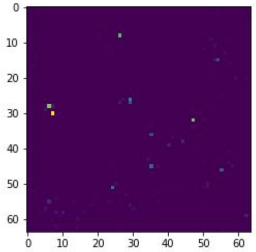


Fig 2. (b) Signal

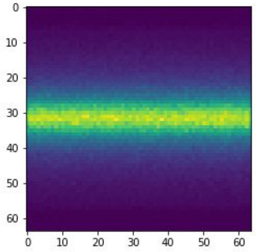


Fig 1. (c) Background averaged

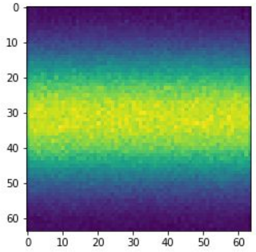


Fig 1. (d) Signal averaged

Fig. 1 Plots (a) and (b) shows the distribution of energy in the Calorimeter* in a randomly selected event. (c) and (d) show the normalized average distribution over the entire dataset as simulated in arXiv:1711.03573.

Weights yet to be added*

- The readings from calorimeter expressed as a 2D image are to be classified as signal or background.
- The Signal represents SUSY - signals.
- In this particular example, the simulated data is binned into 64X64 image.
- Each pixel value represents the energy in the corresponding location of the calorimeter.

Why Neural Networks?

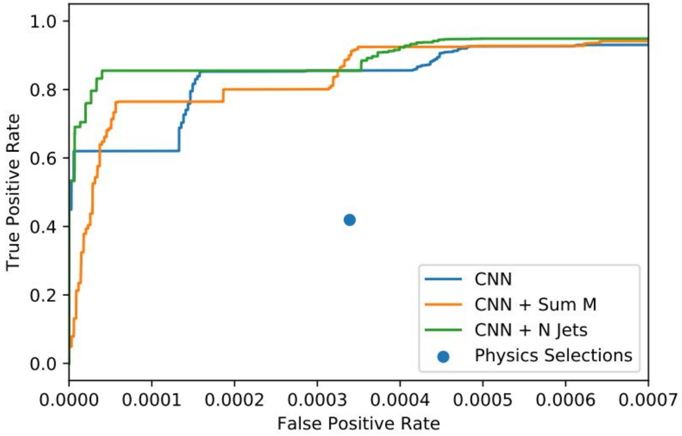


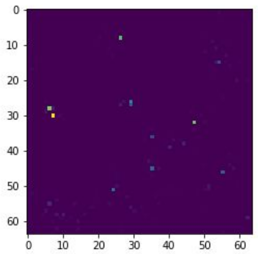
Fig 2. Results of implementing NN

CNN provides better results on lower level calorimeter data than BDTs on higher level physical parameter!

Result from arXiv:1711.03573.

Moving on to OPUs...

Quick Recap on OPU

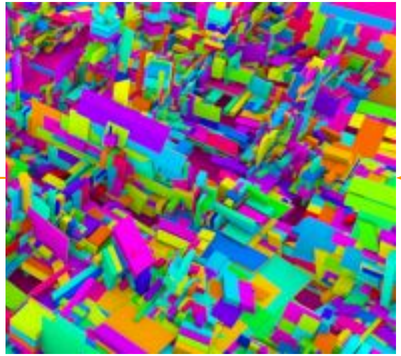


IMAGE

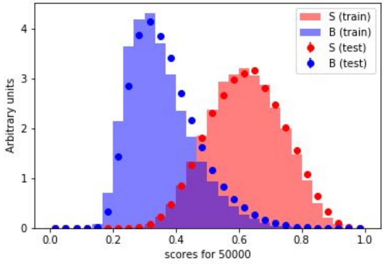
0
1
1
.
.
.
1
0
0

Binary data

OPU



Random Features



Score

Quick Recap on OPU

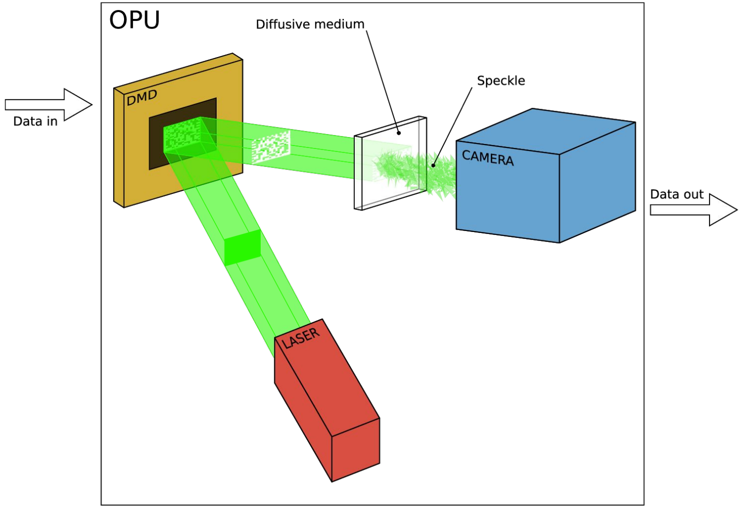


Fig 3. OPU construction

- Convert non-linearly separable data into linearly separable. In the end train with a linear model like ridge regression.
- Speed up the process
- Consume less power

Quick Recap on OPU

Ridge Regression

$$\hat{\beta}^{\text{ridge}} = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

$$= \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \underbrace{\|y - X\beta\|_2^2}_{\text{Loss}} + \lambda \underbrace{\|\beta\|_2^2}_{\text{Penalty}}$$

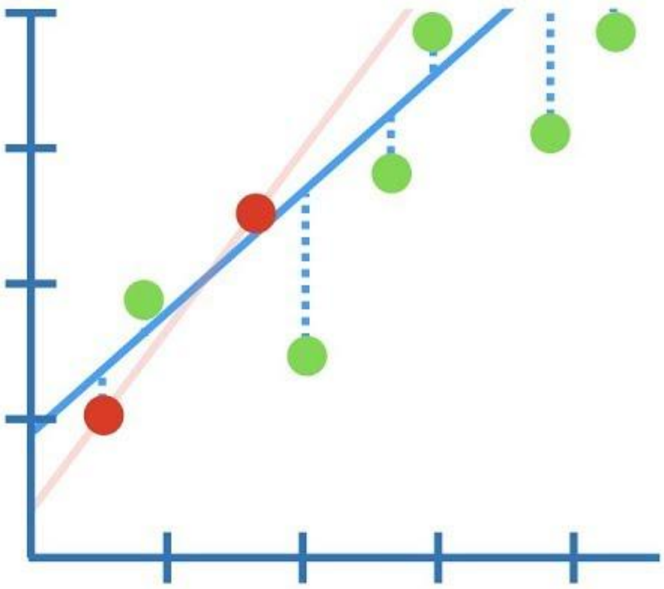


Fig 4. Ridge regression

Data Binarization on OPU

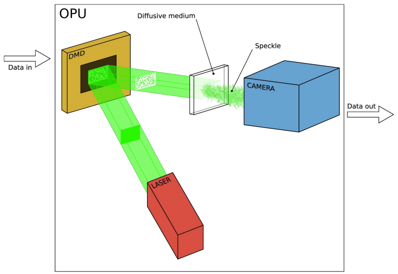


Fig 3. OPU construction

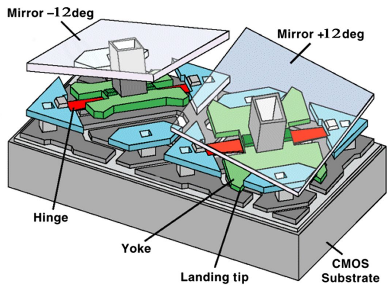


Fig 5. Micromirror on DMD

The construction of the OPU uses a Digital Micromirror Device (**DMD**) to encode data into optical signals.

A DMD consists of an array of micromirrors that can be switched between **two possible** states representing “on” and “off”. In the “on” state, photons are directed towards the diffusive media.

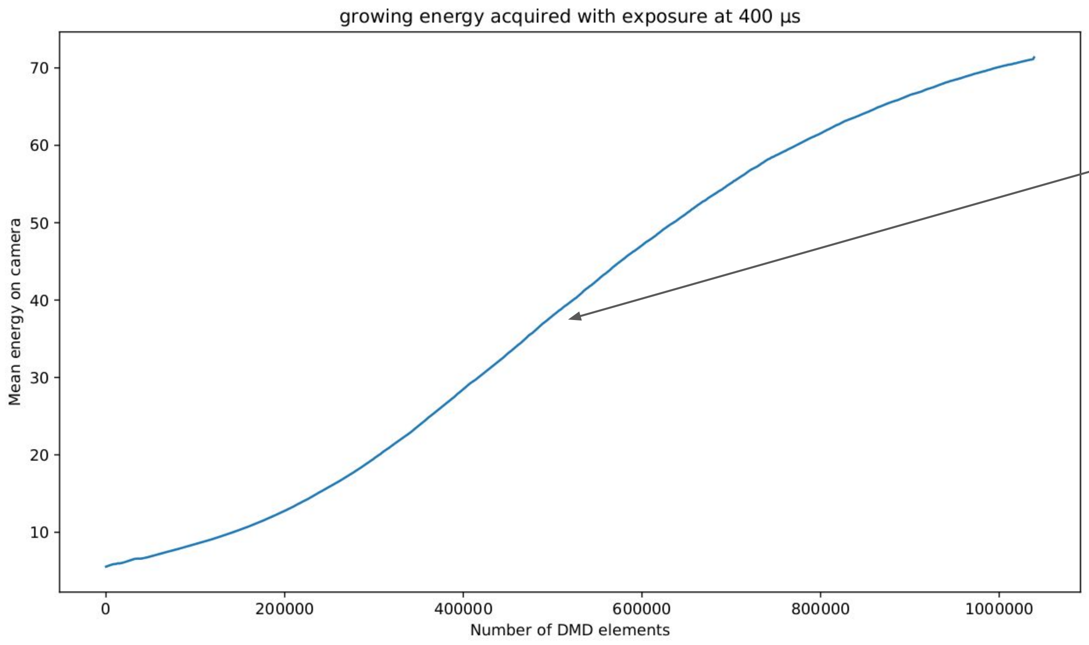
The DMD thus requires the input data to be encoded in **Binary** and the size of input is restricted by the DMD size - 912x1140

Encoding Schemes

The data encoding scheme has a great impact on the final performance and a number of different techniques can be applied for the purpose.

- Autoencoder
- Threshold encoder
- Binning

Encoding Schemes



Near optimal performance?

Thanks to the LightOn team for the graph!

Random Features

Light from the DMD is passed through a diffusive medium. The intensity of light recorded by the high-resolution camera represents the entries of the Random feature matrix.

The **number of features needs to be optimized** not just problem-to-problem it also depends upon the input size.

Exceeding the optimal value leads to **overtraining** of the model.

Random Features

```
since = time.time()
n_components = 5000 #32*32
opu_mapping = OPUMap(n_components=n_components)
train_random_features = opu_mapping.transform(X_train_bin[:50000])
test_random_features = opu_mapping.transform(X_test_bin)
projection_time = time.time() - since
print('Time taken by RP on OPU: {:.4f} s'.format(projection_time))
```

Mapping 50,000 images (already converted into binary) such that each image now has 5000 random features.

Preparing our data

Encoding scheme	AUC
Autoencoder	.86
3 bits 8 bins	.92
Binary Threshold	.932
3 bits 4 bins	.938

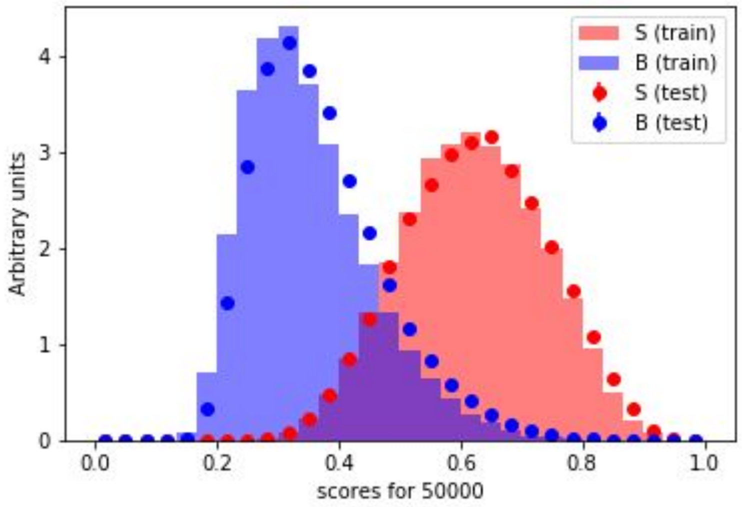
000
001
011
111

Equal sized buckets

Autoencoder model is only trained not tested!

More reason to binarize by hand!

Results



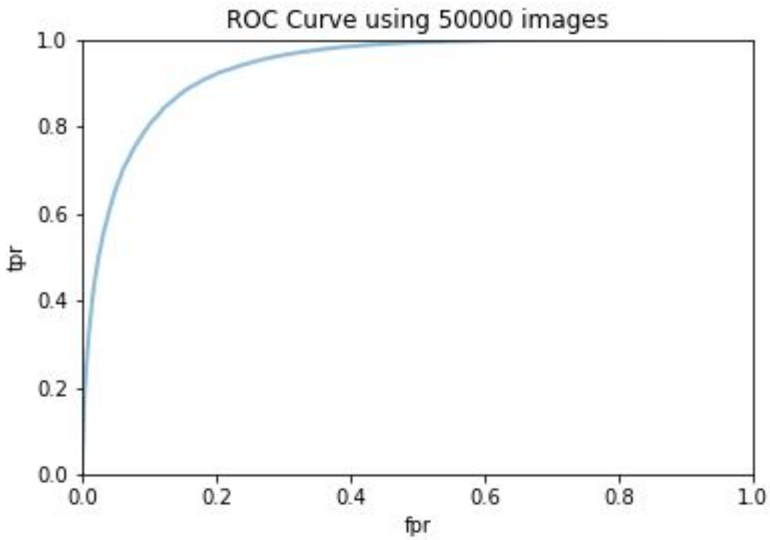
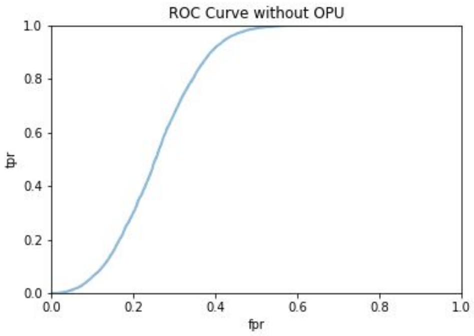
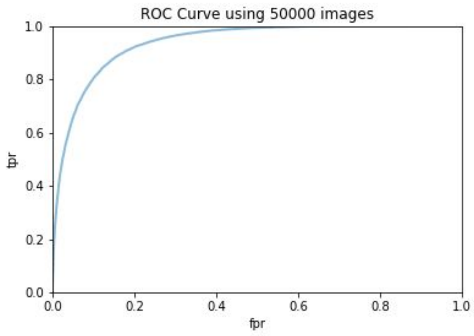
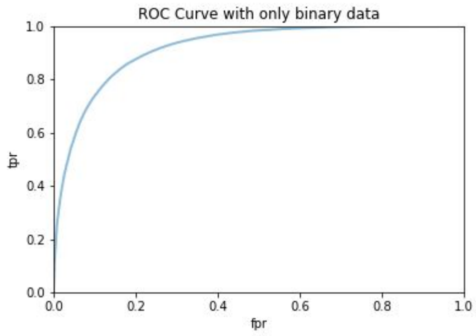


Fig 7 (a) shows the plot for scores trained on 50000 images while (b) shows the corresponding ROC Curve

Results







Demonstrating performance of ridge on raw data, OPU random features and binary data respectively

Subsampling of Feature

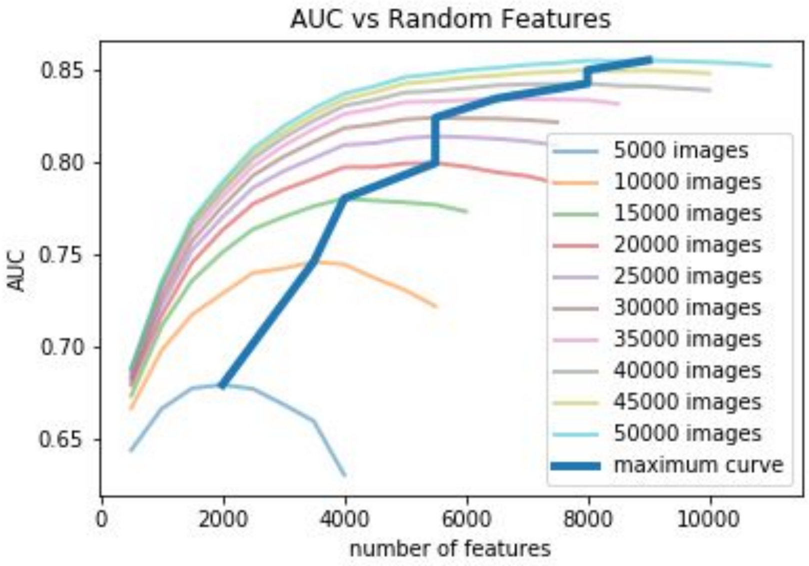


Fig. 8 (a) Results of encoding with autoencoder

Each input feature contains information about all input parameters!

The graph can be drawn by sub-sampling (pick first few features out of a larger OPU mapped space)

Random Features

- The optimal number of random features increases as the number of input images increases (almost linearly).
  Ridge regression soon runs out of memory!
- Beyond the optimal value of the number of random features, the AUC decreases (overtraining)
- Threshold encoding worked much better even when using less number of features.

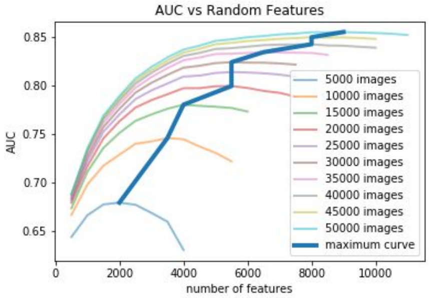


Fig. 8 (a) Results of encoding with autoencoder

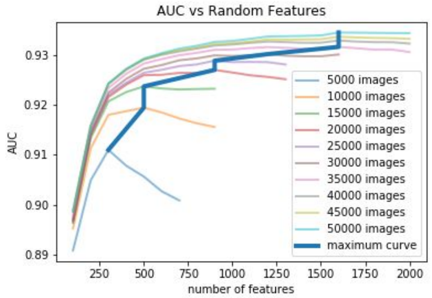
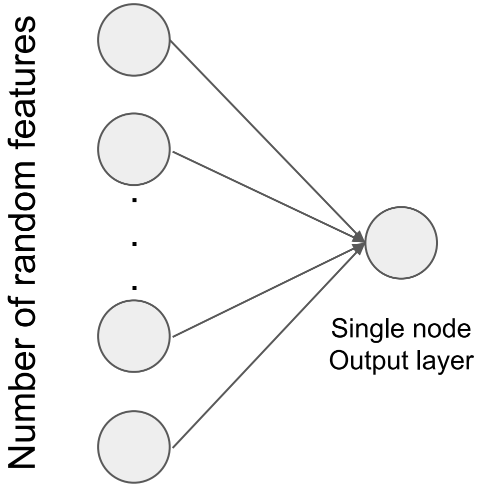


Fig. 5 (b) Results of encoding with Binary threshold encoder

Fig 8 (a) and (b) shows the variation of AUC as the number of input features is varied for different encodings

How do we scale?

Linear Neural Networks



- Scalability: train in mini-batches
- Higher sensitivity to regularization
- Performance comparable to sklearn's ridge regression

- However, the data required preprocessing such as taking sqrt() and minmax scaling in our test case

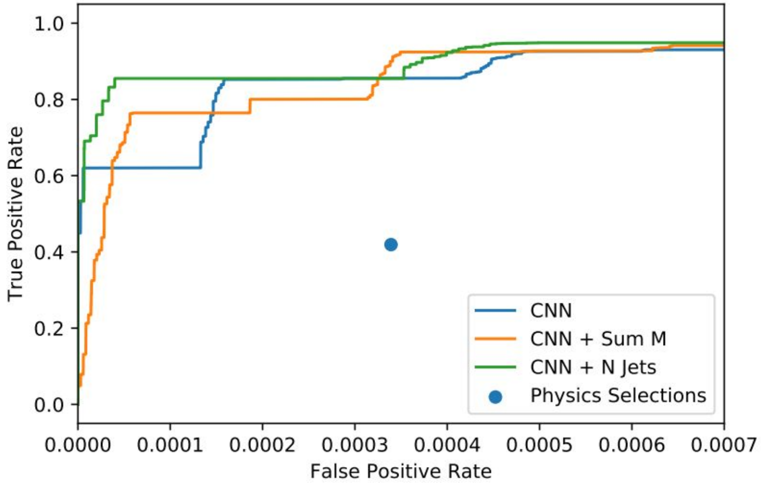
Linear Neural Networks

```python
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(train_random_features.shape[1], 1)


    def forward(self, x):
        x = self.fc1(x)
        #x = torch.sigmoid(x)
        return x
```

```python
model = Net()
criterion = nn.MSELoss()
optimizer = optim.Adam(model.parameters(), lr=5e-5, weight_decay = .1,amsgrad=True)
model.train()
for num_epochs in range(10):
    counter = 0
    for inp, lbl in data_loader_train:
        logits = model.forward(inp)
        loss = criterion(logits, lbl)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        if(counter%10 == 0):
            print('epoch {}, loss {}'.format(num_epochs+1, loss.item()))
        counter = counter+1
```

Comparing Results



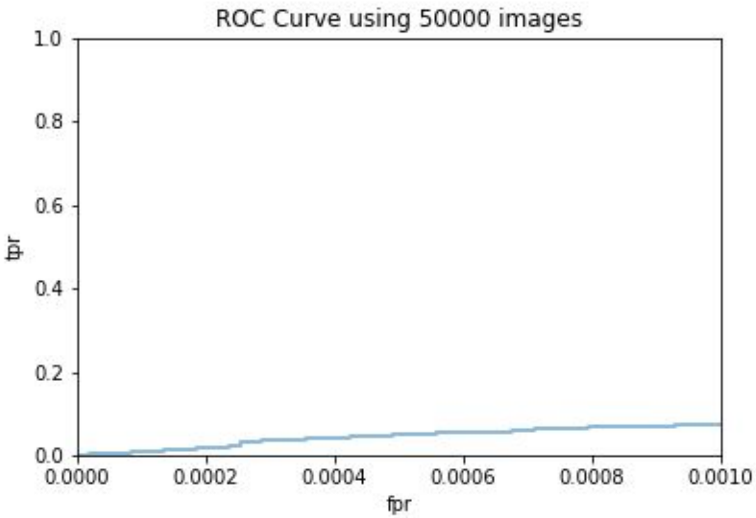


Fig 9 (a) shows the performance of Neural Networks on the images (b) shows Performance of the OPU

CAVEAT: Weights yet to be added! we learned about them from berkeley team yesterday)

is the comparison fair?

Is it Fair to Compare?

While comparing our results with that of the paper we need to consider the following:

- Weights yet to be added!
- We refrained from Transfer Learning!
- It takes ≃ 10 min to train model with 3,00,000 images by converting them to linearly separable data!
- Training done on only 50000 images (1/8th of the total data)

Particle Tracking!

Tracking by definition is clustering!

How to fit data into DMD?

DMD size is a bottleneck.

How to cluster?

Mapping back to the original points?

We lose information about the original points during the mapping!

Thank you