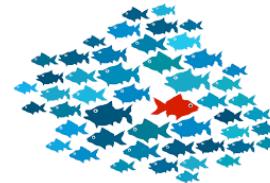


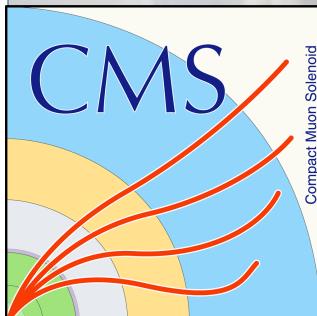
Anomaly detection for ECAL DQM



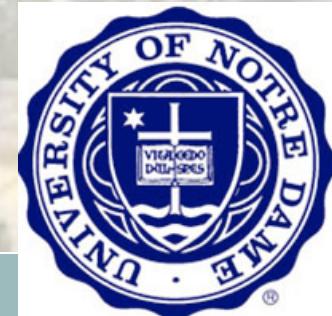
Nabarun Dev¹
for
ML4DQM team

28/02/2018

¹ University of Notre Dame

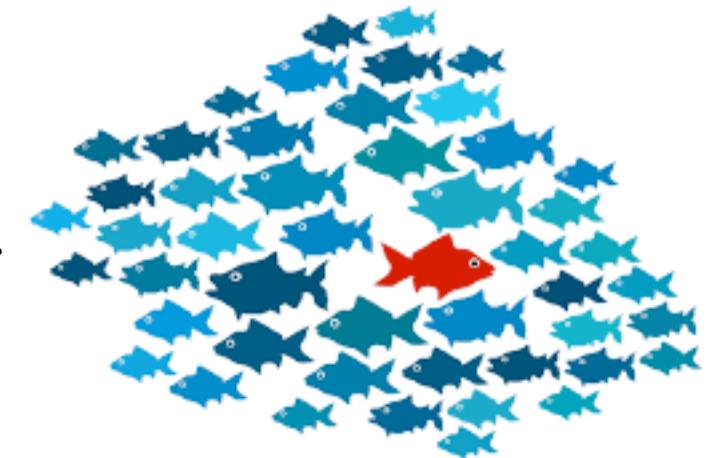


CMS ML forum meeting



Introduction

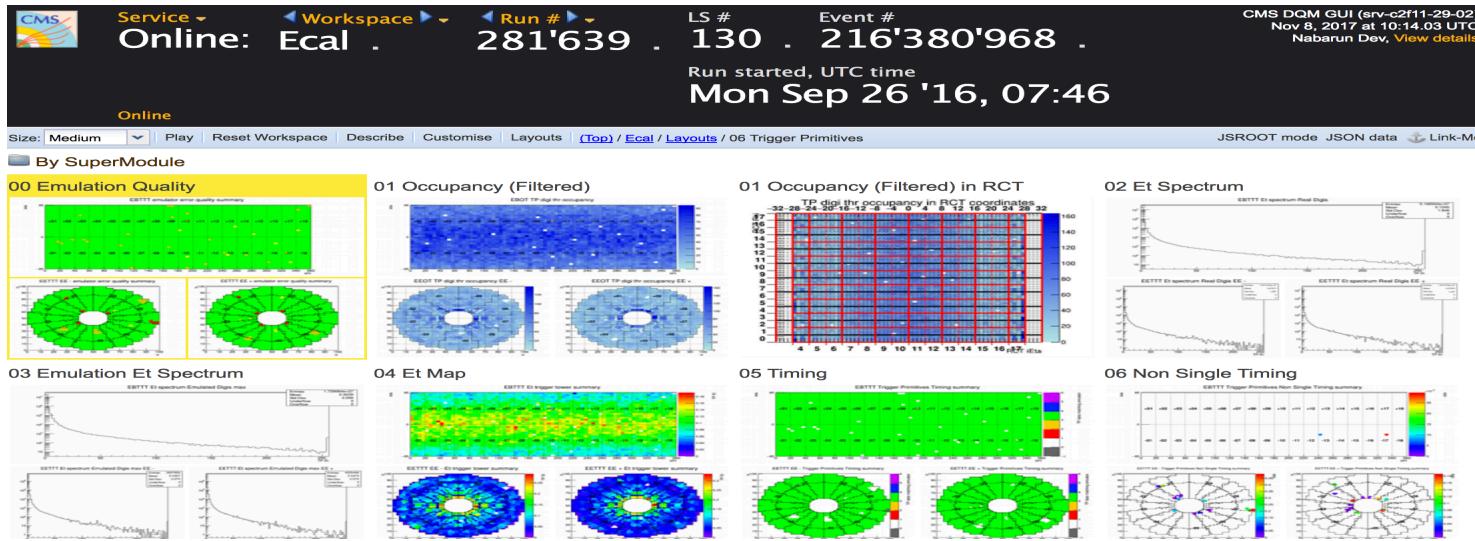
- DQM (data quality monitoring) system is an important tool to ensure high quality data-taking for analyses purposes. It is used both online and offline.
- In real time data quality is currently assessed by looking at a dashboard containing a set of histograms which are compared to a reference set of histograms according to certain set of instructions.
- By spotting **anomalies** in these images (plots/histograms) it is possible to identify problems that appear in the detector, flag poor quality data and/or take steps towards fixing these issues.





ECAL DQM

- The ECAL DQM consists of a set of several histograms that help us monitor the running condition of the ECAL subdetector. DQM GUI shown below:



- The histograms are usually redrawn every lumisection. By monitoring these the shifter is able to spot problems in real time.



Avenues of improvements

- Although the DQM system has been performing well over the years there are areas which can be potentially improved.
- The number of plots to monitor can be overwhelmingly large which can cause delay in spotting a problem or cause a transient problem to be overlooked.
- Lot of manpower is needed to constantly monitor these systems during data-taking.
- There is a possibility that the monitoring decisions can vary from shifter to shifter.

Machine Learning techniques can help reduce these issues.



DISCLAIMER: WORK IN PROGRESS



Current Strategy

CAN YOU FIND ODD SMILEY FACE OUT?



- Use only normal instances (semi-supervised learning) to train an autoencoder (input is mapped to output, the system learns to reconstruct the input with minimum loss) [1] to minimize the loss function.

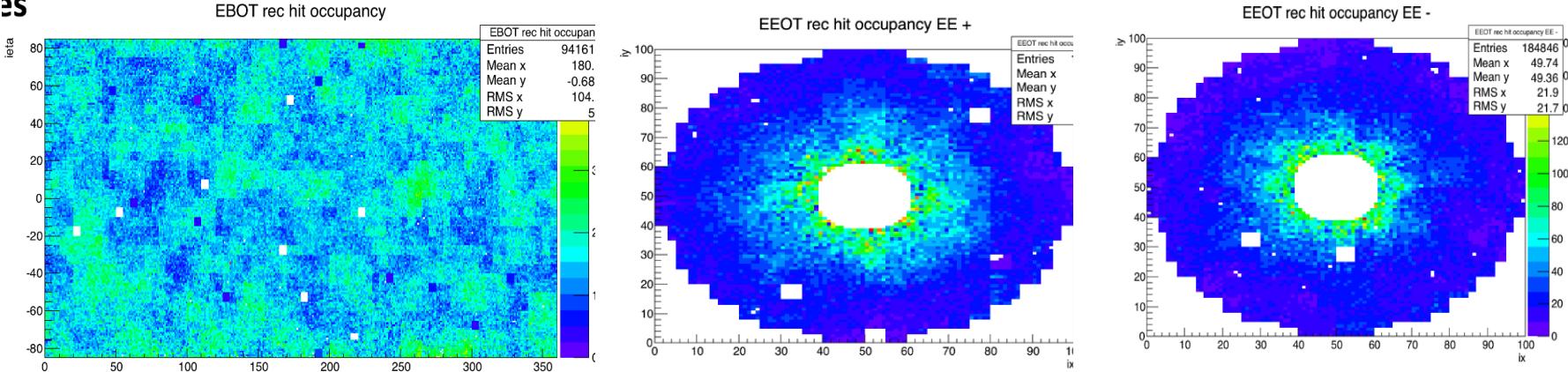
- The loss can then also serve as a discriminant:
 - Good instances should be reconstructed with low loss
 - Bad instances should be reconstructed with higher loss

[1]:<http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/> ,
<http://www.deeplearningbook.org/contents/autoencoders.html>

Dataset

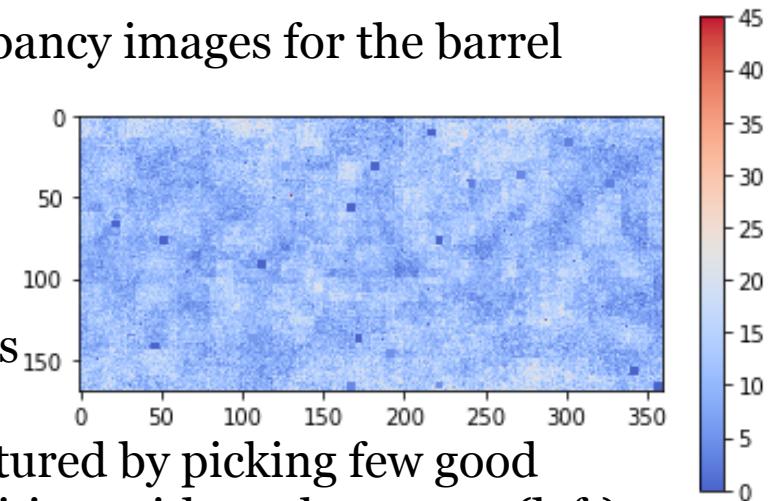
- Current dataset consists of ~40000 samples from 2016 data from lumisections marked as good. ([2016 goldenjson](#) , [CMSSW_9_2_11](#))
- The dataset ([SingleElectron/RAW](#)) is processed to emulate the online DQM running conditions and produce one sample (set of images) per lumisection ([40k sets of images from 2016, 70k from 2017](#))
- The current image set per sample consists of rechit occupancy and timing plots: one for barrel and one each for both endcaps.

28

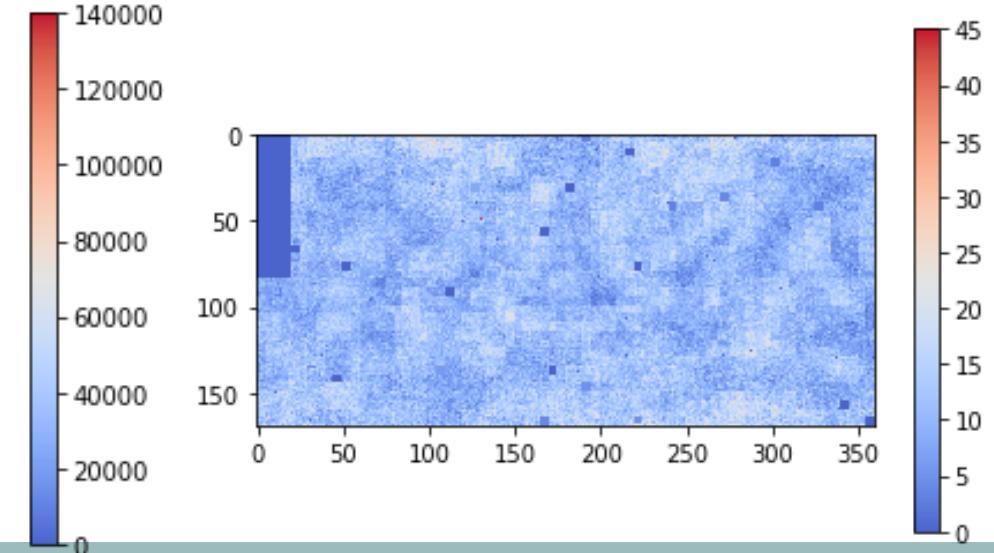
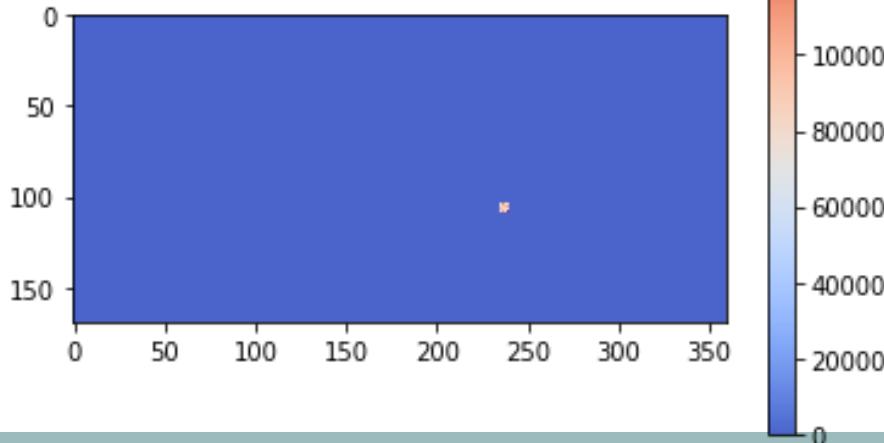


Dataset

- The following study only uses the rechit occupancy images for the barrel as normal images:

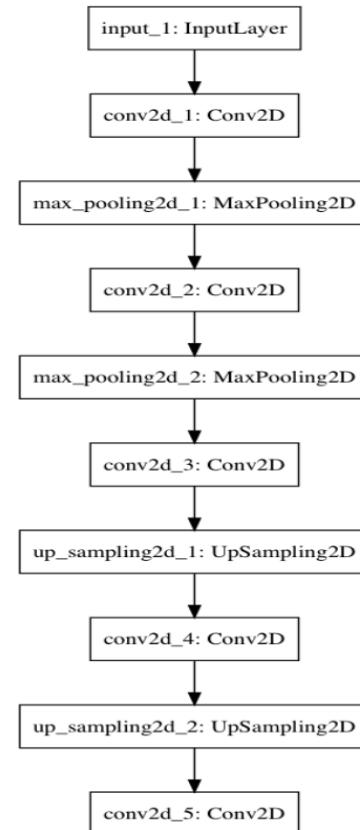


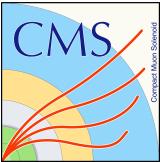
- Was able to acquire very few anomalous images
- Two types of anomalous images were manufactured by picking few good images and artificially inserting at random positions either a hot tower (left) or a missing module (right)



Architecture

- Models with several architectures were trained
- All models so far are (plain) autoencoders with convolutional layers
- Conv2D (6 channels,(5x5) patches) →
 MaxPooling2D(5, 5) → Conv2D (6 ,(4x4)) →
 MaxPooling2D(2, 2) → Conv2D (4 ,(2x2)) →
 MaxPooling2D(2, 3) → Conv2D (4 ,(2x2)) →
 UpSampling2D(2, 3) → Conv2D (4 ,(2x1)) →
 UpSampling2D(2, 2) → Conv2D (6 ,(4x4)) →
 UpSampling2D(5, 5) → Conv2D (6 ,(5x5)) →
 Conv2D(1,(5x5))
- All Conv layers are ‘padded’ to keep size of output channels same as input.
- Framework Used:
 Keras library using tensorflow backend

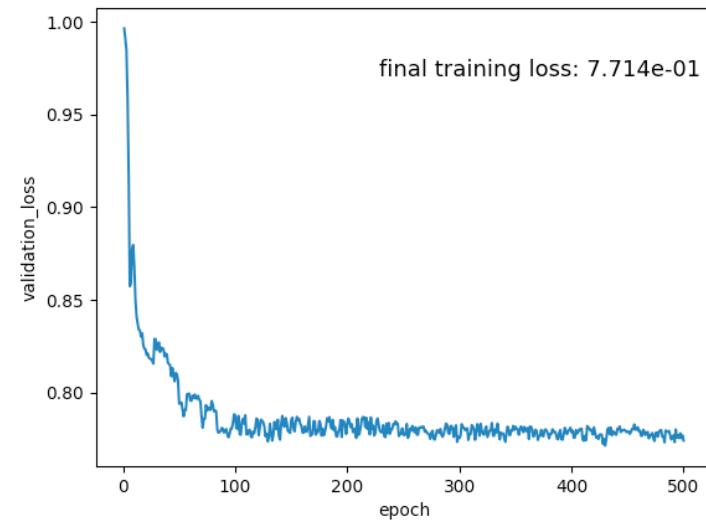
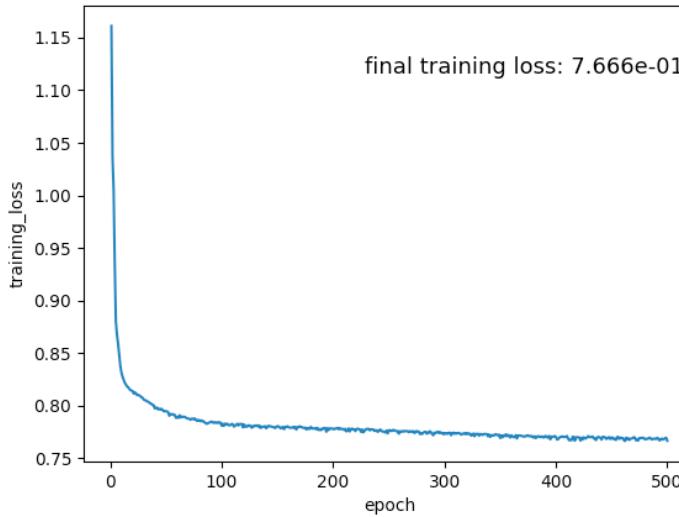




Model 1

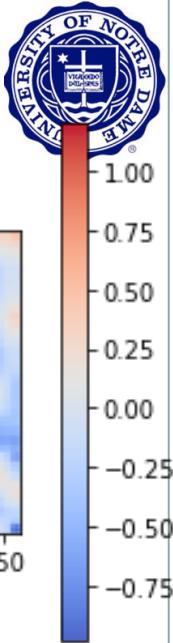
- Preprocessing scale to 0 mean and standard deviation of 1:
$$X = \frac{X - \mu}{\sigma}$$
- Input-to-hidden and all hidden-to-hidden layers have LeakyRelu activations.
- Hidden to output layer has linear activation.
- Optimizer: AdaDelta; Loss : MSE loss.
- Regularization: Dropout (10-15%)

Other optimizers, regularization methods were tried with similar results

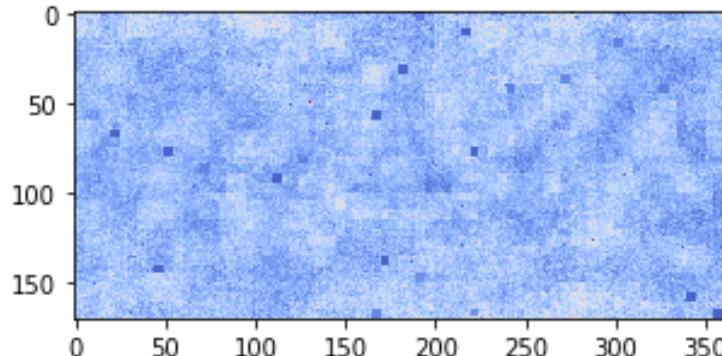




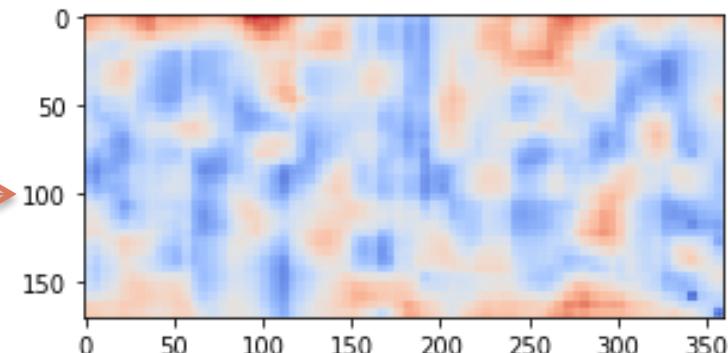
Reconstruction of good LS images: Model I



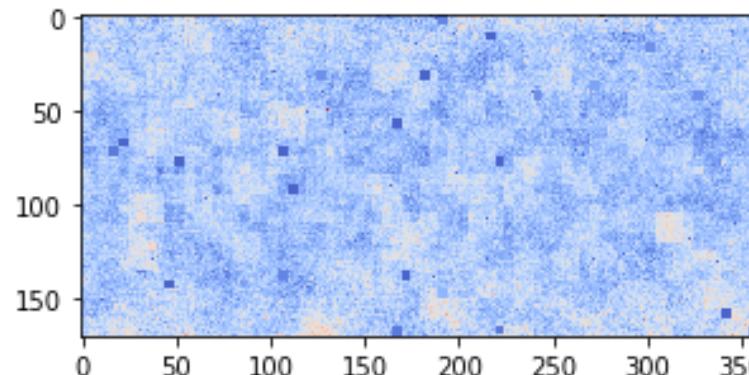
Good Image



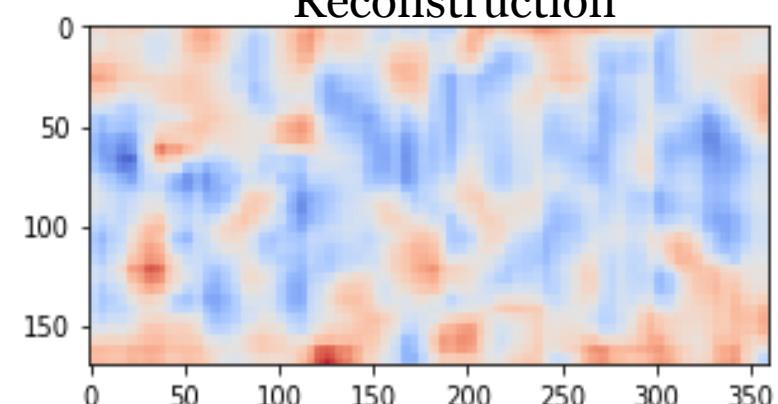
Reconstruction



Good Image



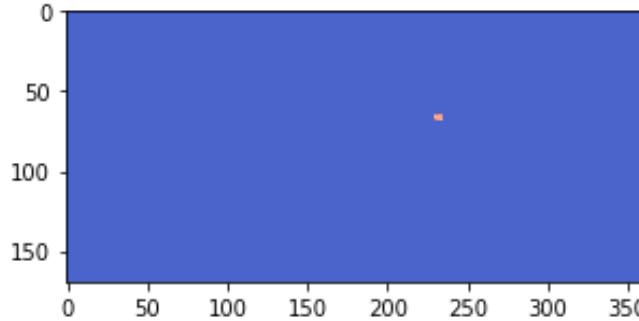
Reconstruction



Code dimension is $\sim 100x$ smaller than input dimension: undercomplete reconstruction

Reconstruction of anomalous images: Model I

Image with Hot Tower



Reconstruction

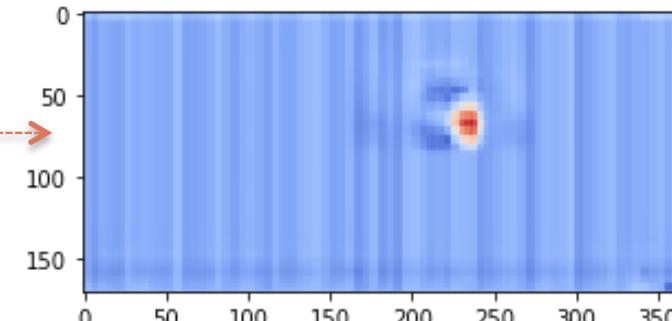
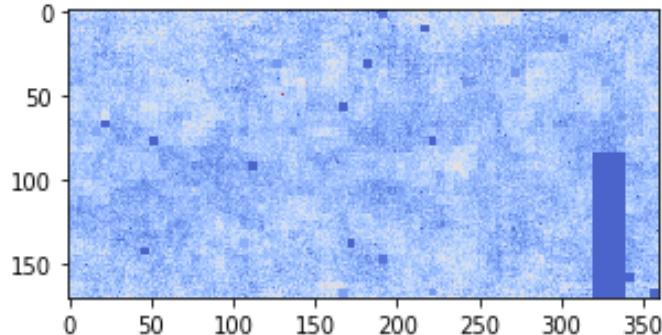
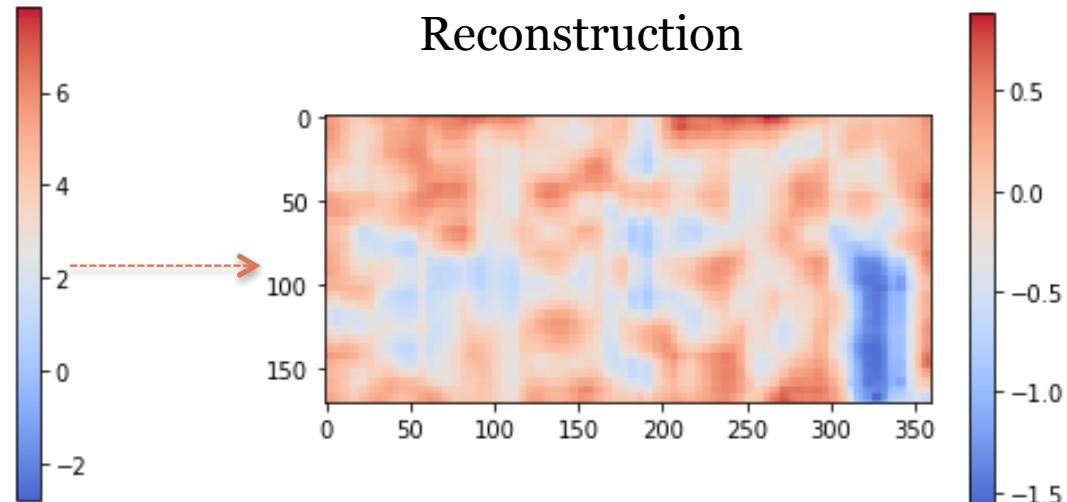


Image with Missing Module



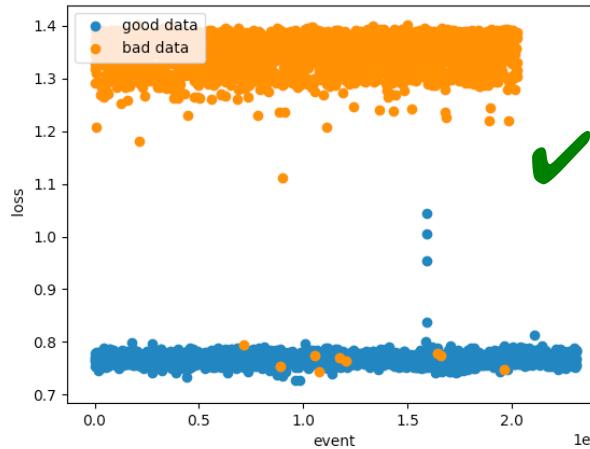
Reconstruction



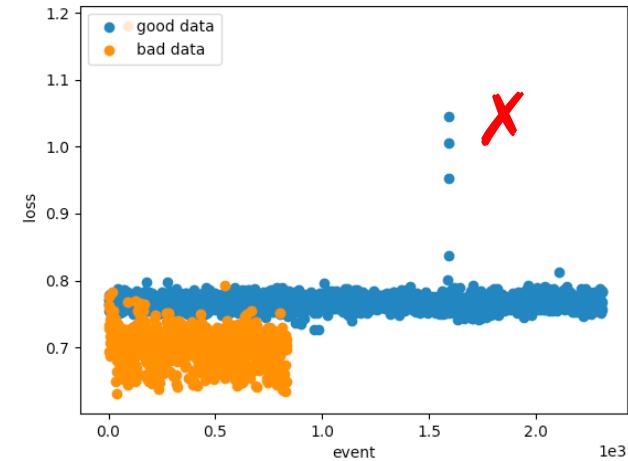
Code dimension is $\sim 100x$ smaller than input dimension: undercomplete reconstruction

Reconstruction loss as discriminant: Model I

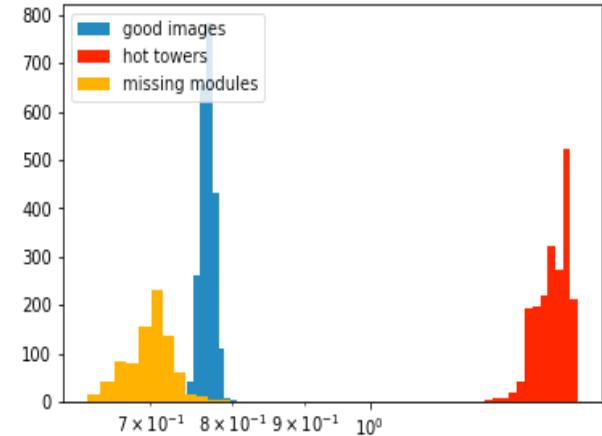
Hot Towers V.
Good



Missing Modules V.
Good



- works well with hot towers: the higher reconstruction loss can be seen as coming from the 'spread' around the hot tower.
- doesn't work with missing modules: It seems to reconstruct the missing module quite well and there is minimal spread around it with well defined edges.

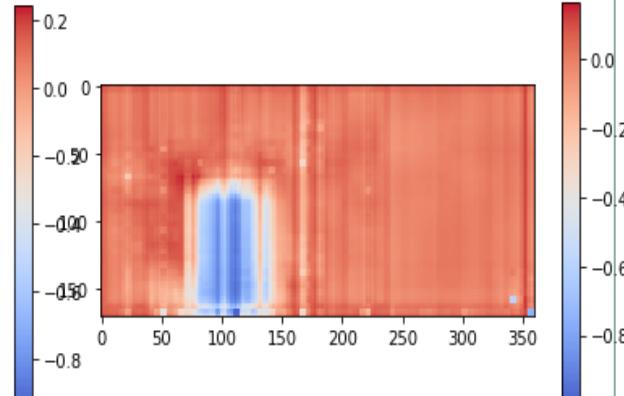
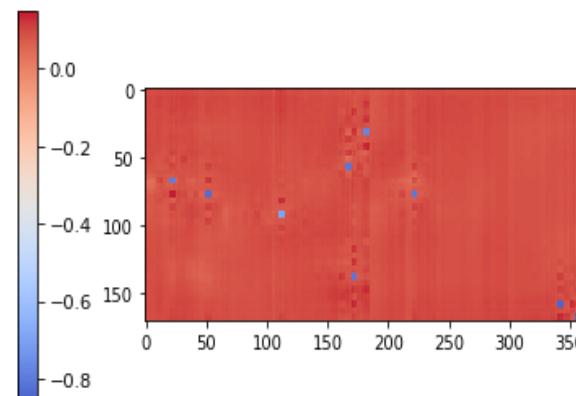
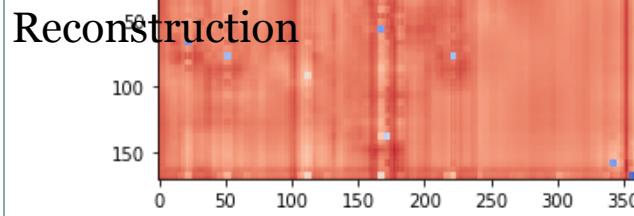
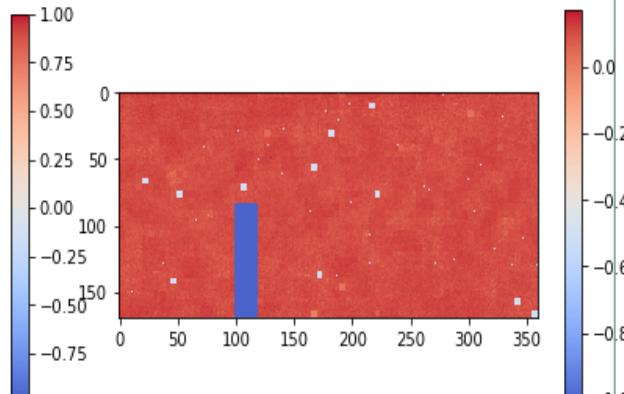
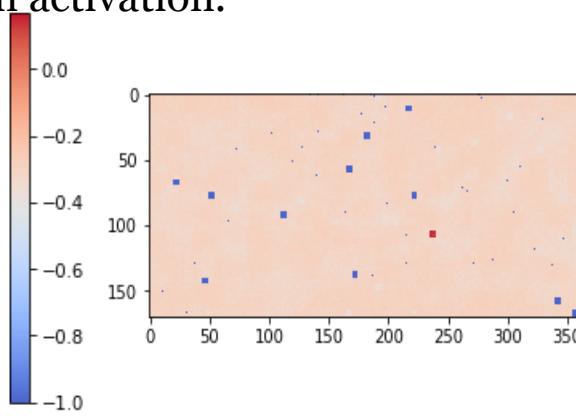
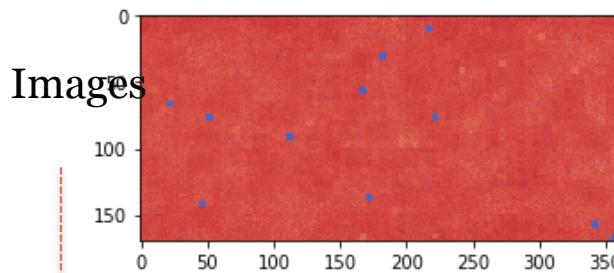




Model 2

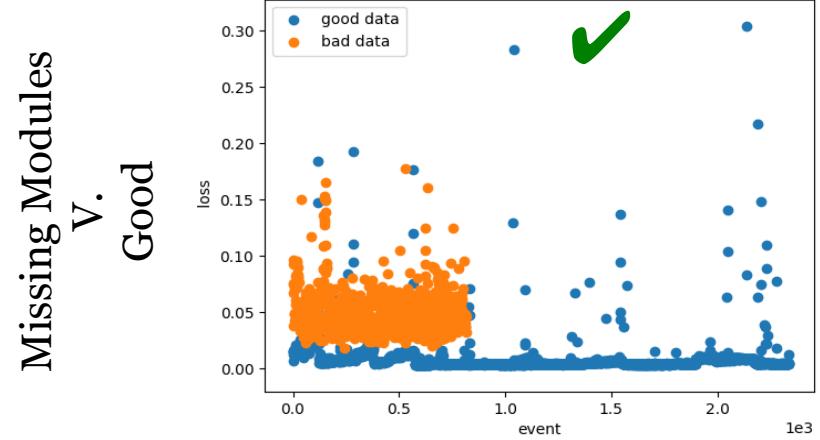
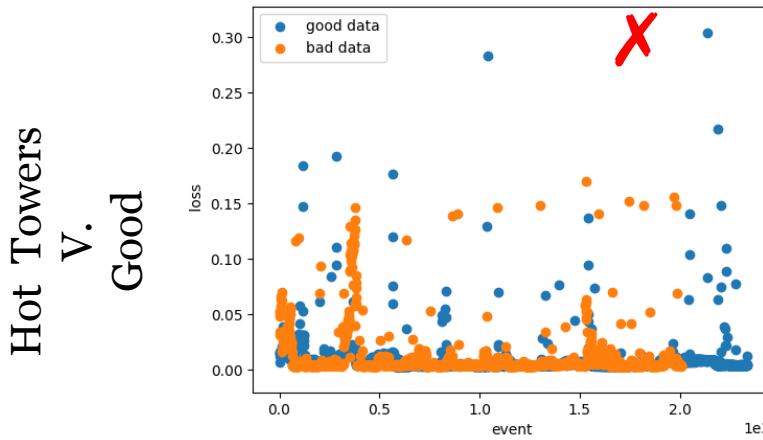


- Preprocessing : log followed by max_absolute_scaling
- Hidden to output layer has tanh activation.



- Unable to reconstruct the hot tower at all.
- Blurs out the edges of the missing module into neighboring areas.

Reconstruction loss as discriminant: Model II

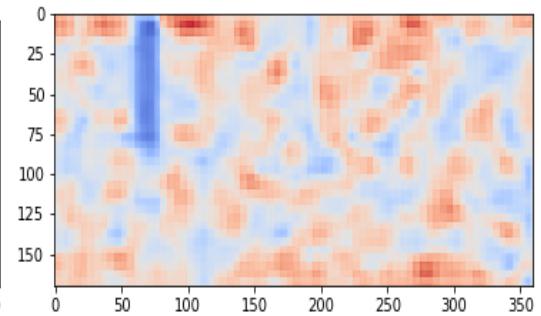
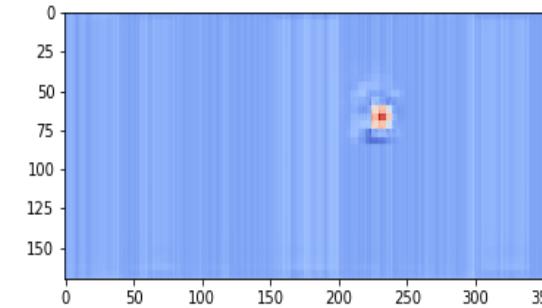
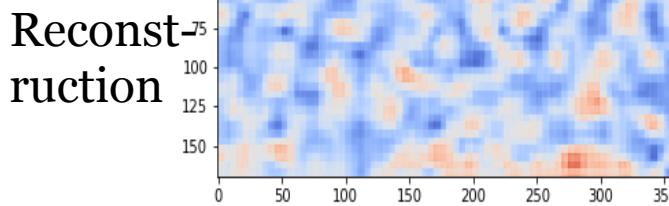
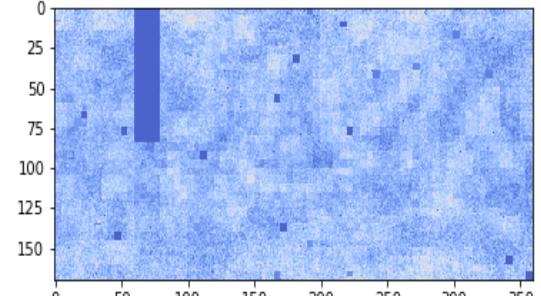
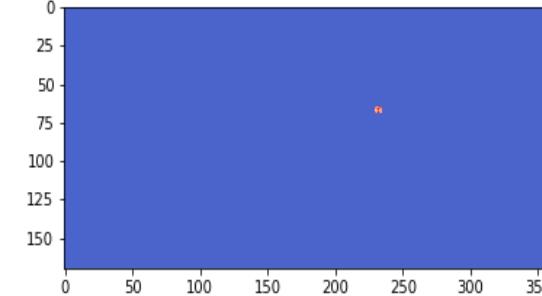
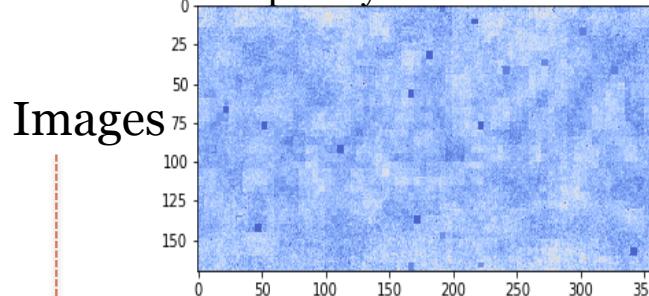


- It is unable to reconstruct the hot tower at all. It seems to uniformly 'warm' up the whole image rather. But the reconstruction error (sadly and understandably) is similar to that of normal images as it doesn't really have any handle to blow up. A better function as discriminant could help here
- The reconstruction blurs out the edges of the missing module into neighboring areas. This leads to a higher loss..

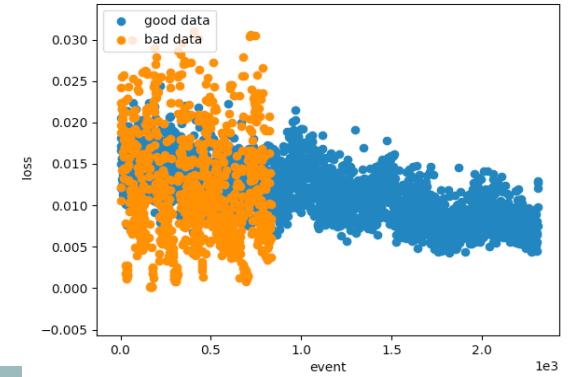
Other strategies

- Preprocessing : $(x - \text{mean})/\sigma$ followed by MaxAbsScaling
- Hidden to output layer has tanh activation.

- Preprocessing : log followed by max_abs_scaling
- Hidden to output layer has tanh activation.

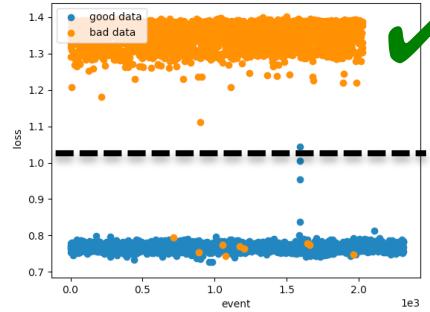


- Reconstructs too well, or at least reconstructs anomalous images as well as good ones. This doesn't help as we are not looking for a generative model
- Loss spectrums of good and bad images not different to allow good discrimination

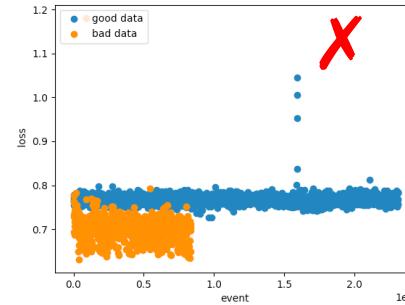


Current best startegy

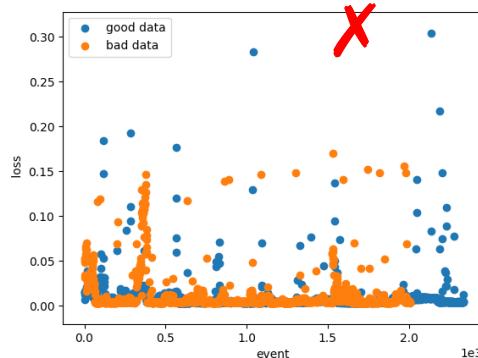
- Current best strategy is to use a mixture of experts: Classify an image as an anomaly if either model 1 or model 2 calls it an anomaly.



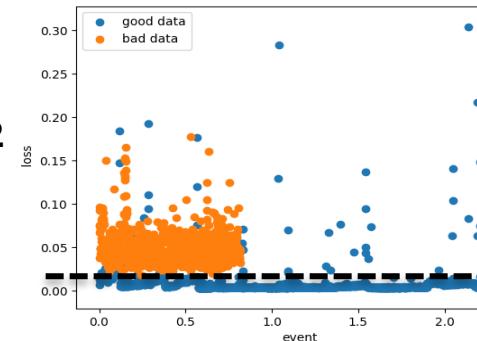
Model 1



OR



Model 2



- Needs lot of improvement!
- A supervised learner based on the same dataset does almost perfectly. (backup)

Next Steps

- Use more sophisticated reconstruction losses; KL divergence as regularization. The magnitude of loss in the regularization term has potential to act as better discriminant.
- Try more sophisticated models like VAEs, DAEs etc.
- Train Model for endcaps
- Use timing information to train a parallel model.

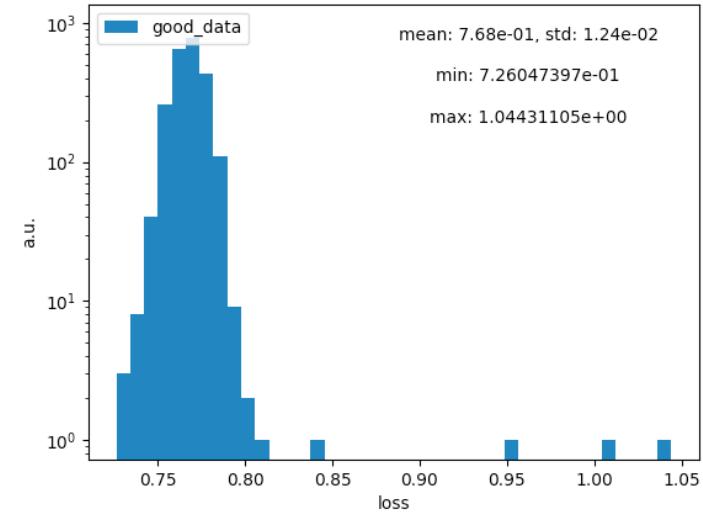
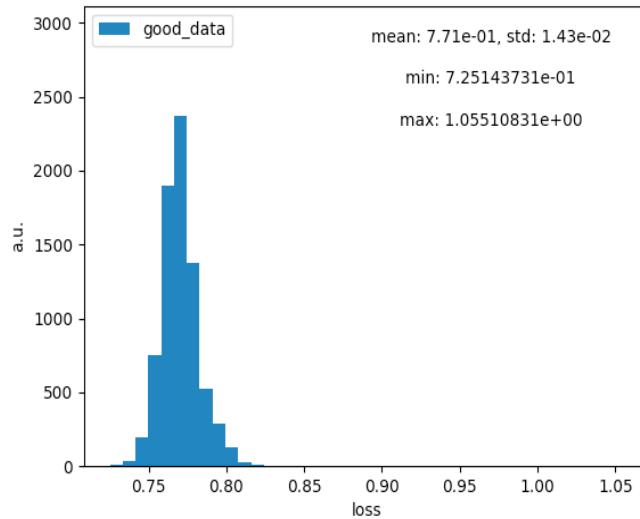
Thank you!

BACK UP



Training /Test Loss Spectrum: Model 1

- Evaluate trained model over (each sample) training and test sets and histogram the reconstruction loss



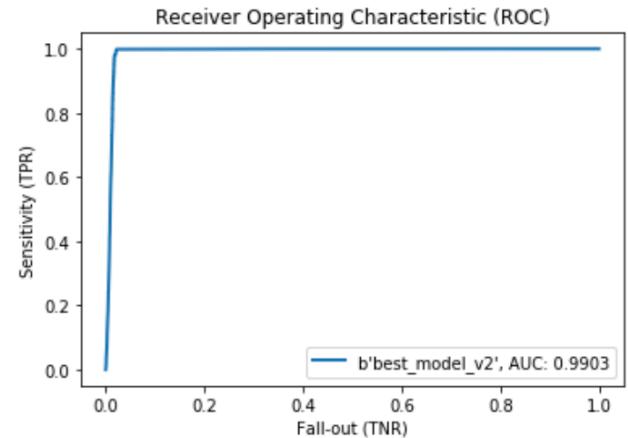
- Training and test sets have similar performance.

Supervised Approach

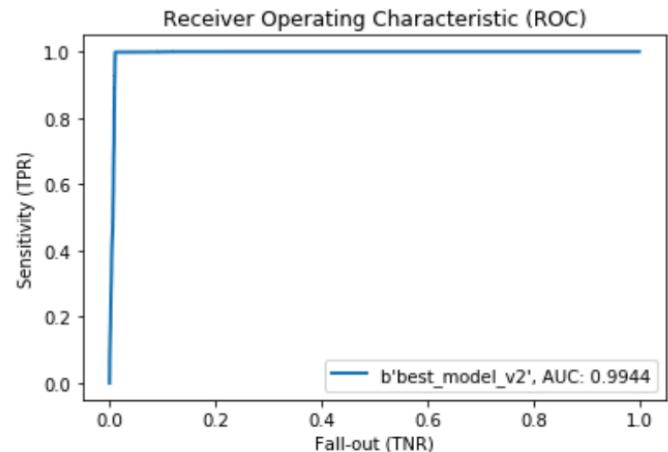
- Looked at supervised approach as an exercise.

- Supervised binary classification
- All kinds of anomalies are one negative class
- Train a net with convolutional layers followed by dense layers.
- Activation: ReLU (except finaly layer has softmax)
- ADAM optimizer
- Binary CrossEntropy loss
- Net architecture in backup
- Regularizers: l2, dropout

Training-validation roc



Testing roc

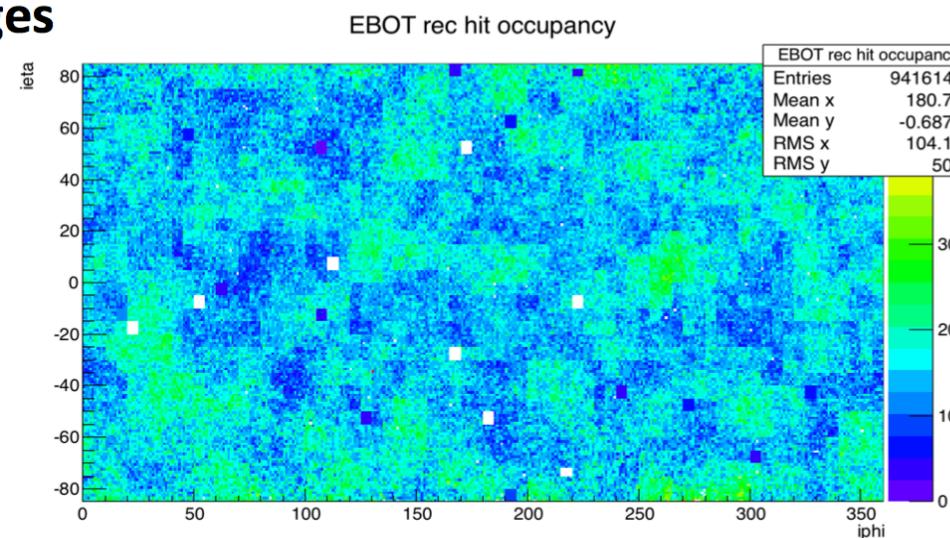


Layer (type)	Output Shape	Param #
=====	=====	=====
===== input_1 (InputLayer)	(None, 1, 170, 360)	o
===== convA_5x5 (Conv2D)	(None, 6, 170, 360)	156
===== leaky_re_lu_1 (LeakyReLU)	(None, 6, 170, 360)	o
===== pool5x5 (MaxPooling2D)	(None, 6, 34, 72)	o
===== convB_4x4_stride2 (Conv2D)	(None, 4, 16, 35)	388
===== leaky_re_lu_2 (LeakyReLU)	(None, 4, 16, 35)	o
===== convC_4x4_stride2 (Conv2D)	(None, 4, 7, 16)	260
===== leaky_re_lu_3 (LeakyReLU)	(None, 4, 7, 16)	o
===== dropout_1 (Dropout)	(None, 4, 7, 16)	o
===== flatten_1 (Flatten)	(None, 448)	o
===== denseA (Dense)	(None, 100)	44900
===== leaky_re_lu_4 (LeakyReLU)	(None, 100)	o
===== dropout_2 (Dropout)	(None, 100)	o
===== denseB (Dense)	(None, 2)	202
=====	=====	=====
===== Total params:	45,906	Trainable params: 45,906
=====	params: o	Non-trainable

Supervised learning net architecture

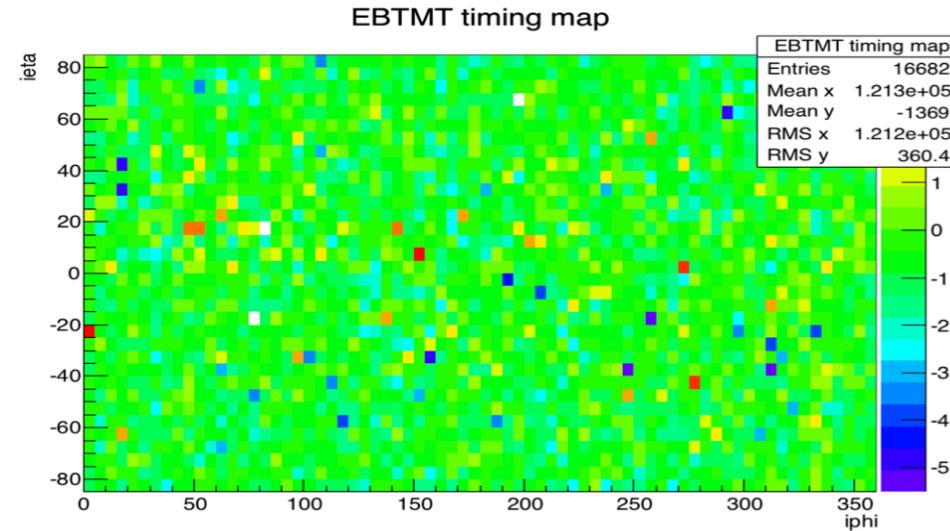
Input Images

170, 360



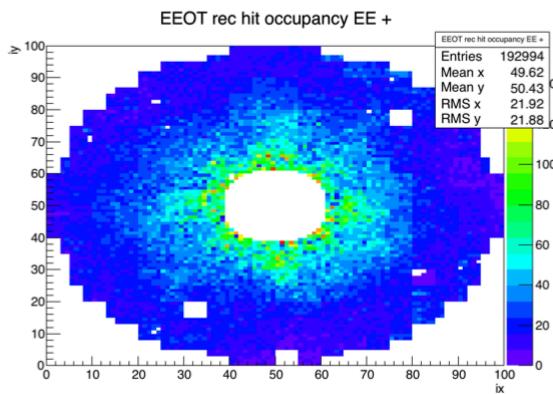
BARREL

34, 72



Input Images

100,
100



ENDCAPS

20,20

