Full Detector Simulation of Pair Monitor and Machine Learning

Ahmed Mustahid

December 6, 2019

Overview

Pair Monitor: Principle

- Many low energy e^+e^- pairs are created during Beam Crossing due to $\gamma\gamma \to e^+e^-$, $e^\pm \to e^\pm e^+e^-$ and $e^+e^- \to e^+e^-e^+e^-$ processes.
- particles with same charge as that of the beam are deflected at a larger angle
- Beam size can be measured using the deflection pattern
- In the current study, incoherent pair particles were simulated by CAIN

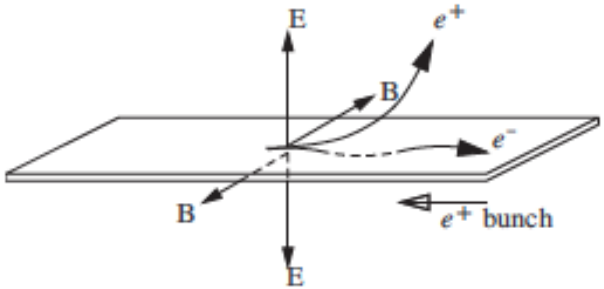


Figure 1: e^+e^- pair creation and deflection of the "same" charged particle[1]

Pair Monitor at ILD

- Pair Monitor has not been implemented inside the ILD framework
- The first layer of BeamCal can fully mimic the Pair Monitor signals
- The midpoint of the first two positions of energy contribution step inside the first layer of BeamCal has been taken to be the signal.
- Detector simulation has been carried out by using DD4HEP package using ILD-l05-v05 configuration

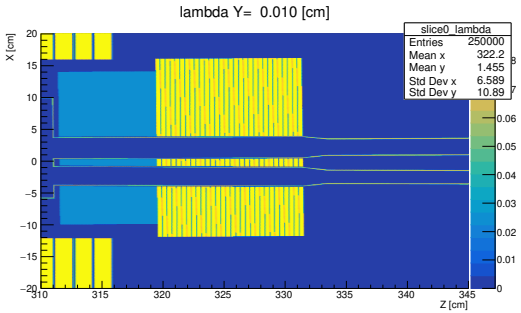


Figure 2: BeamCal view from Y perpendicular region

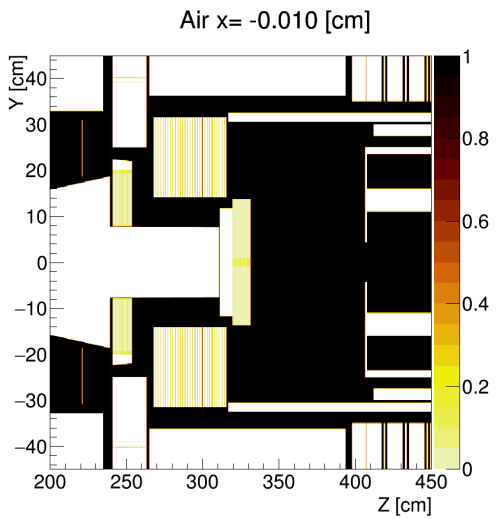
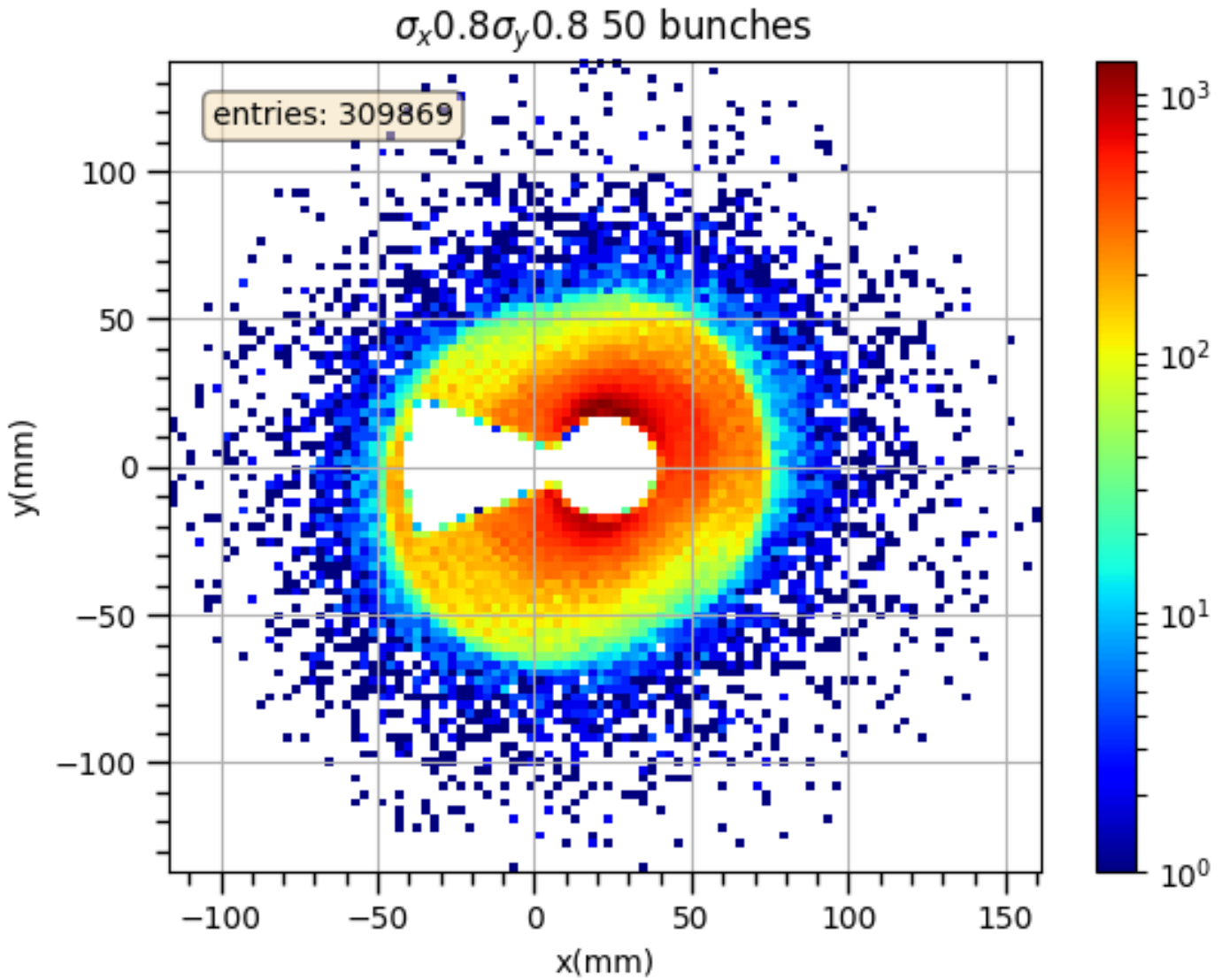


Figure 3: FCal region in ILD framework [2]

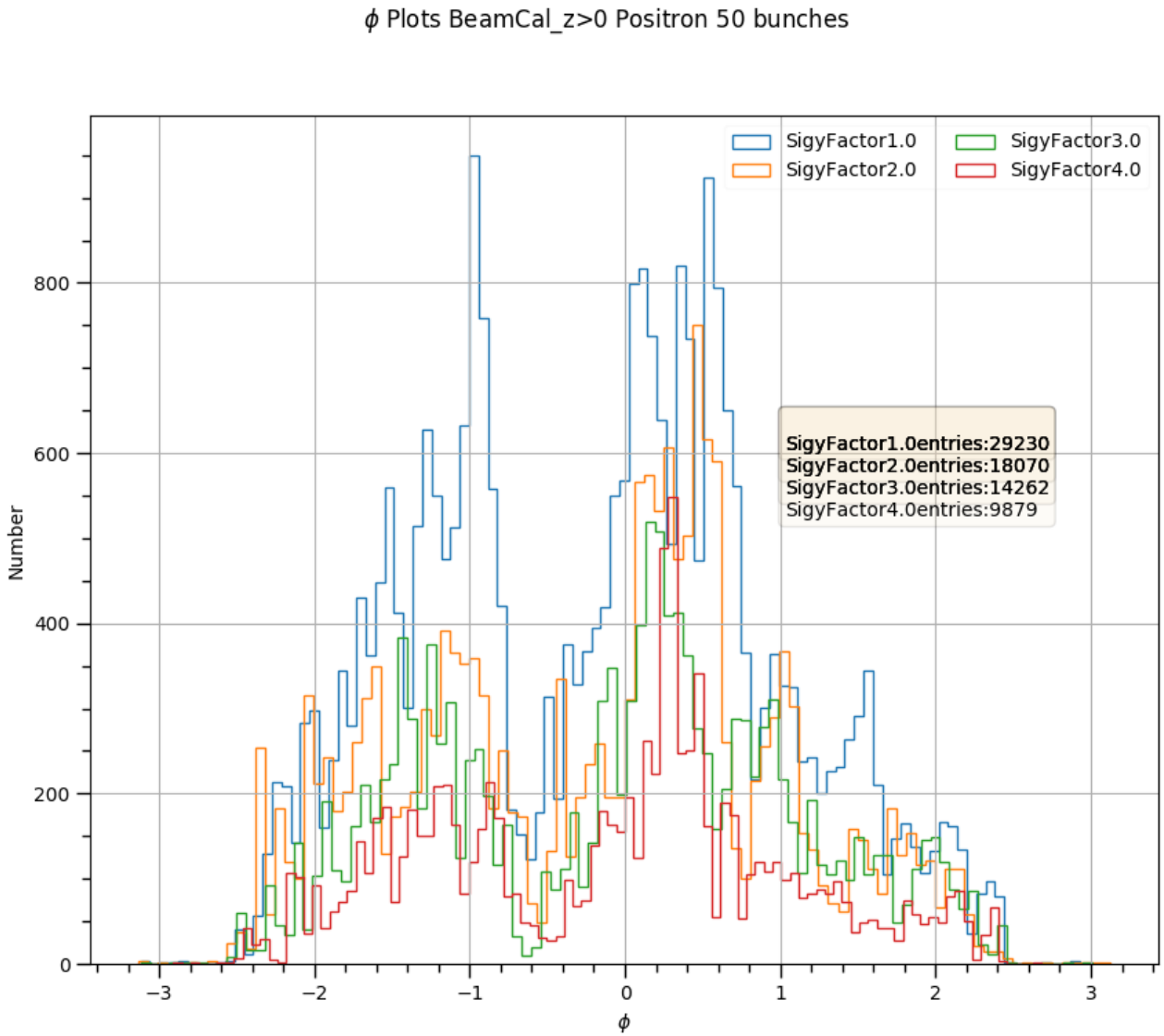
Position Distribution of the energy contribution



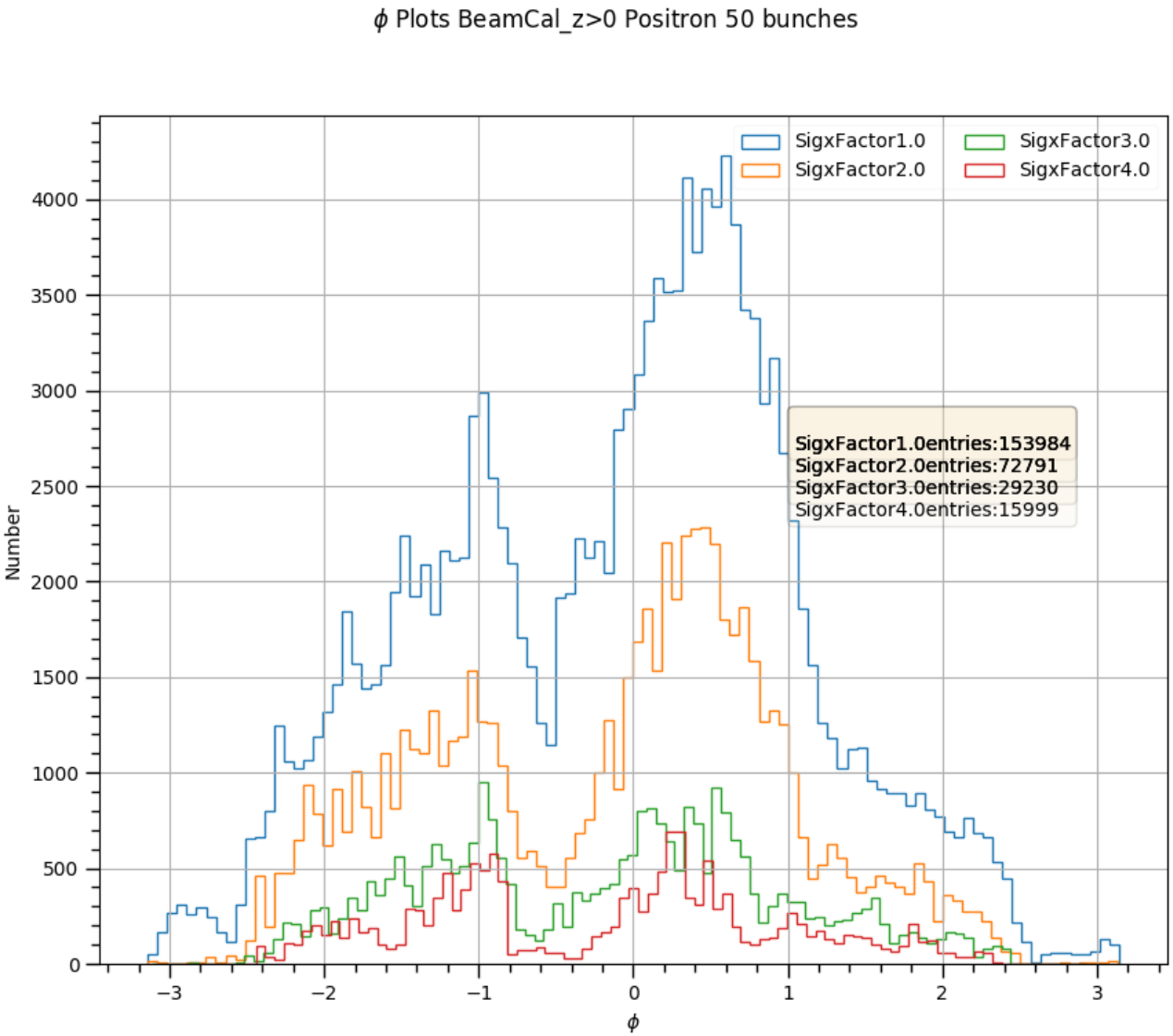
Parameters Used

- Simulation has been carried out for two sets of parameters
- set 1: 4×4 values of σ_x, σ_y: 1,2,3,4
- set 2: 5×5 values of σ_x, σ_y: 0.8,. . . ,1.6 with a step of 0.2

Set 1: ϕ plots for constant σ_x (factor 1.0)

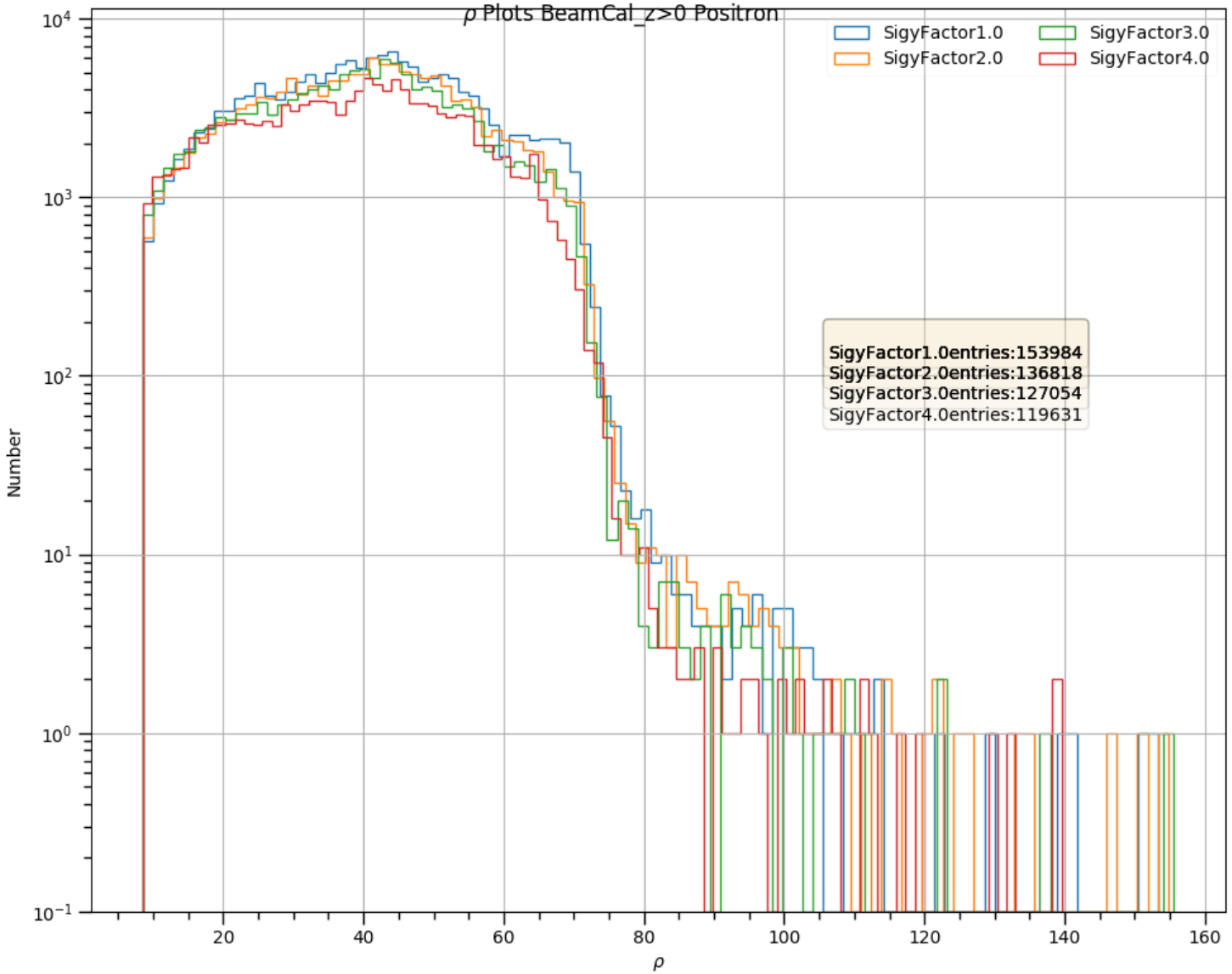


Set 1: ϕ plots for constant σ_y (factor 1.0)

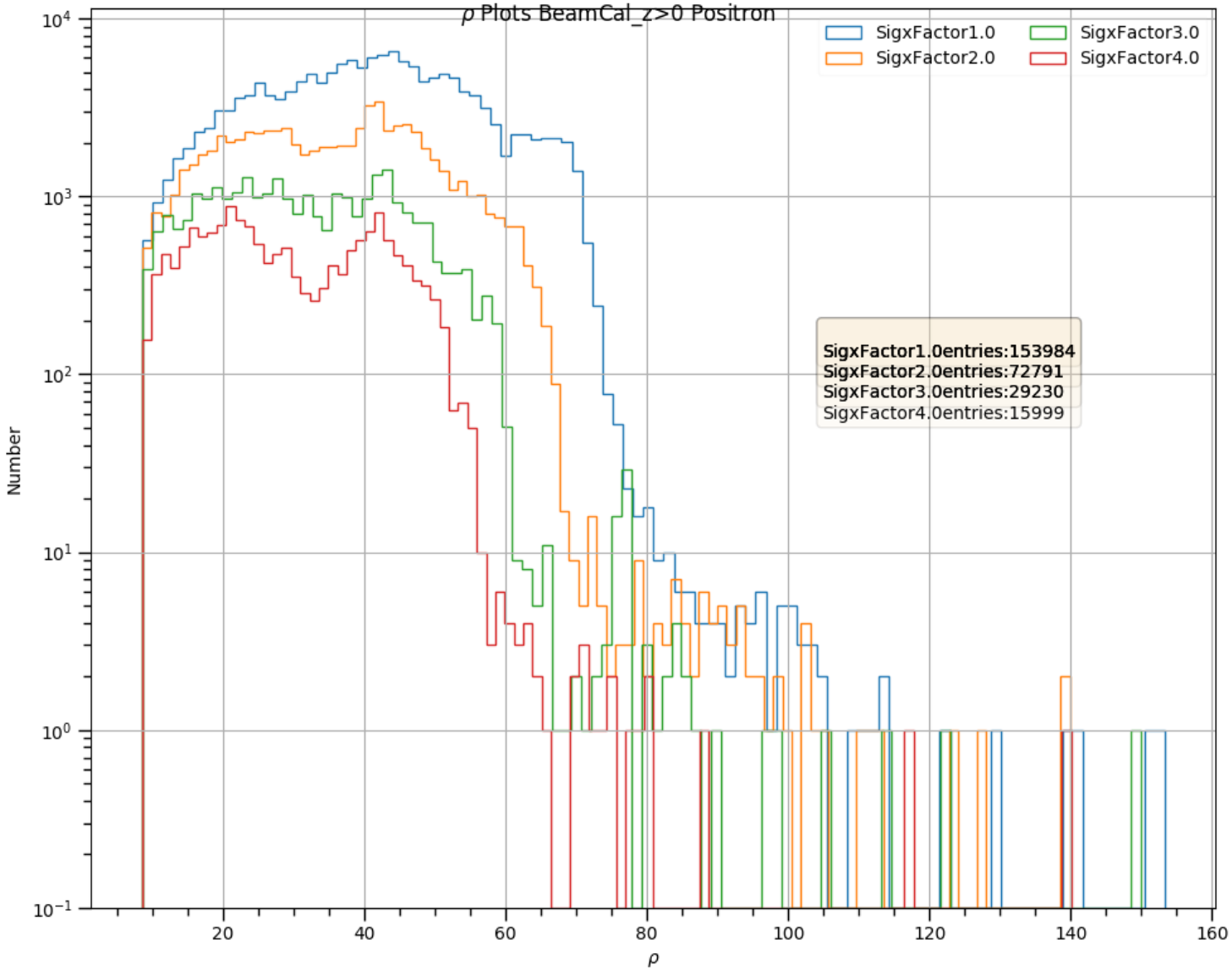


Set 1: ρ plots for constant σ_x (factor 1.0)

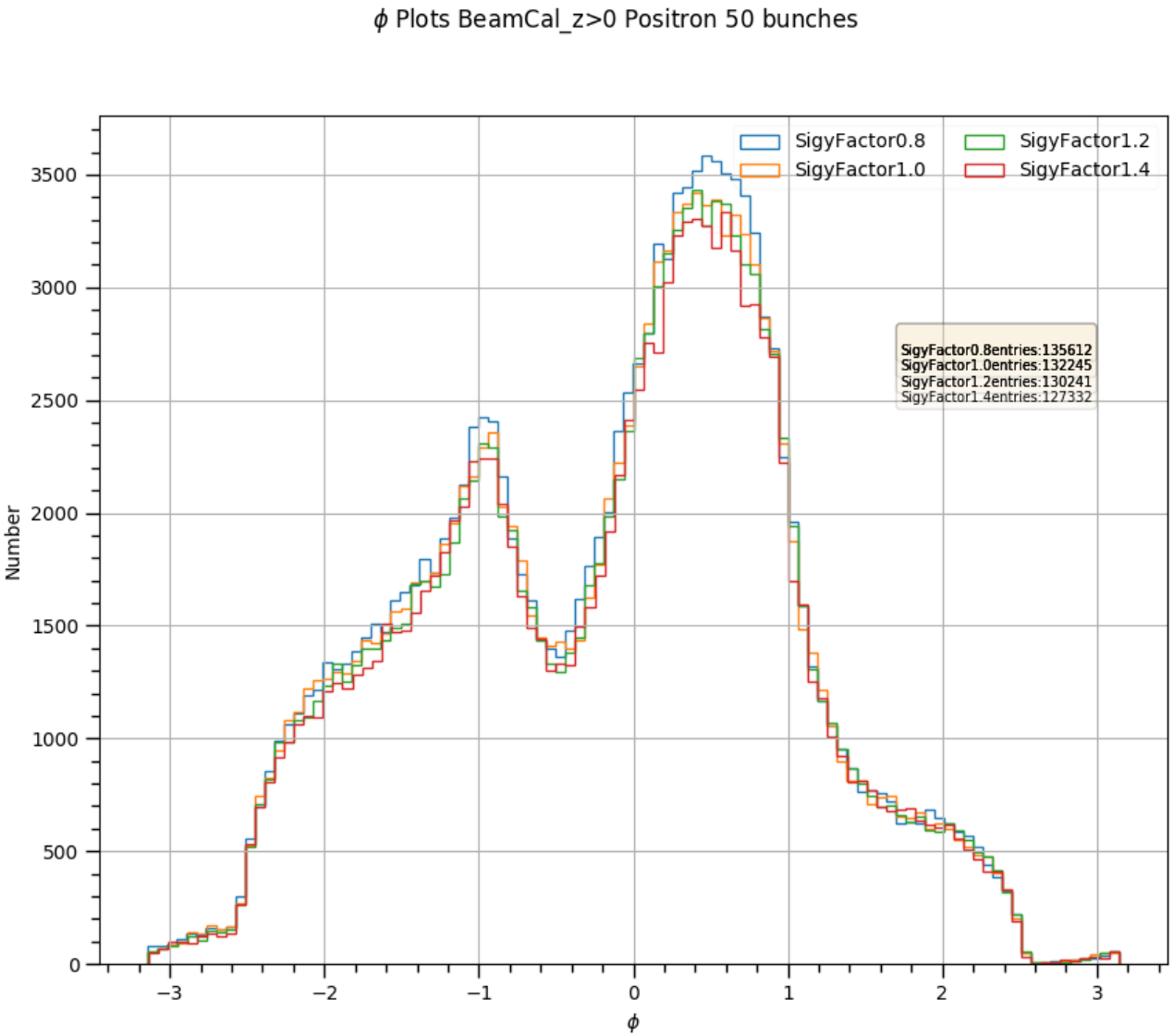
$$\rho = \sqrt{x^2 + y^2}$$



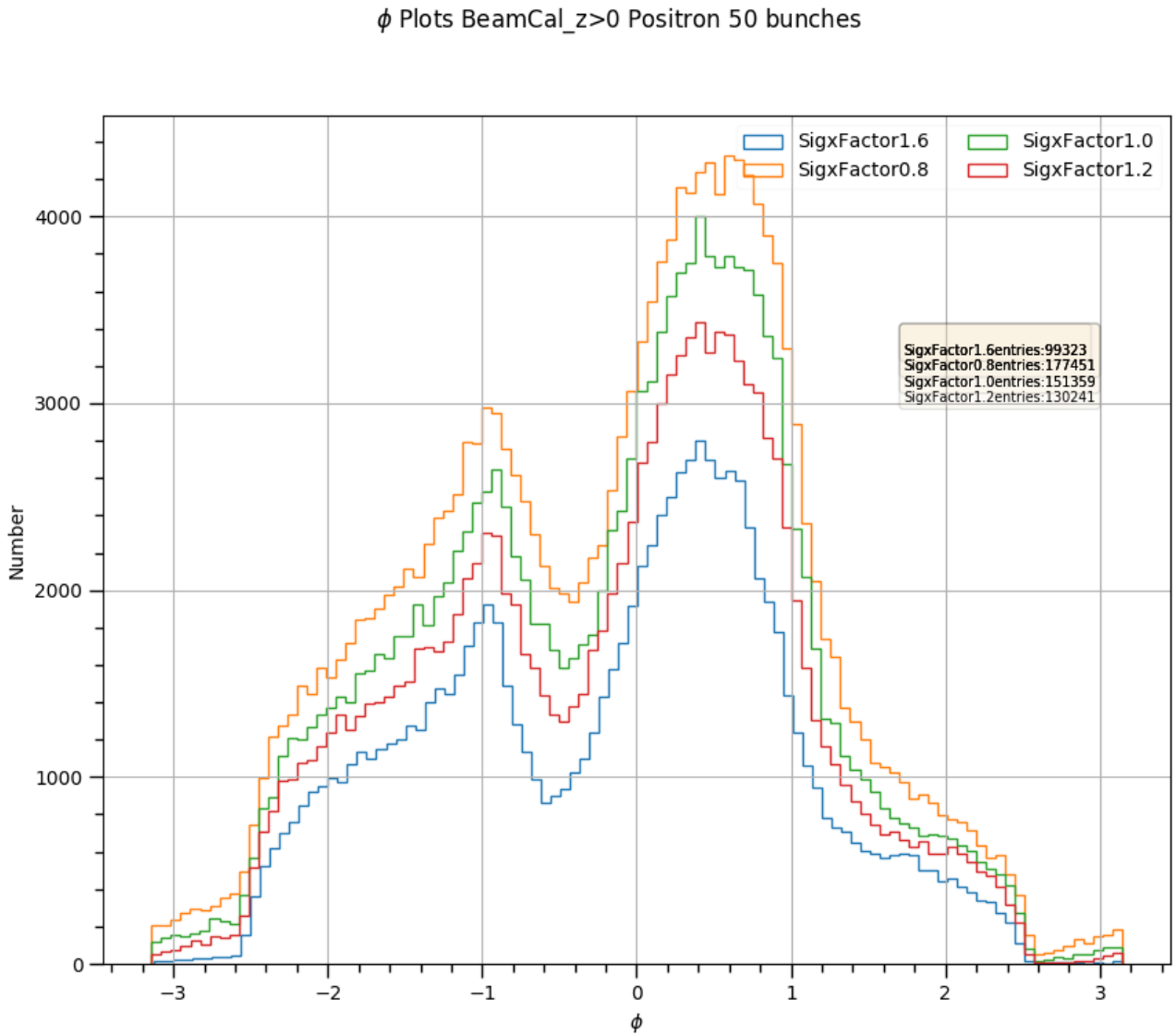
Set 1: ρ plots for constant σ_y (factor 1.0)



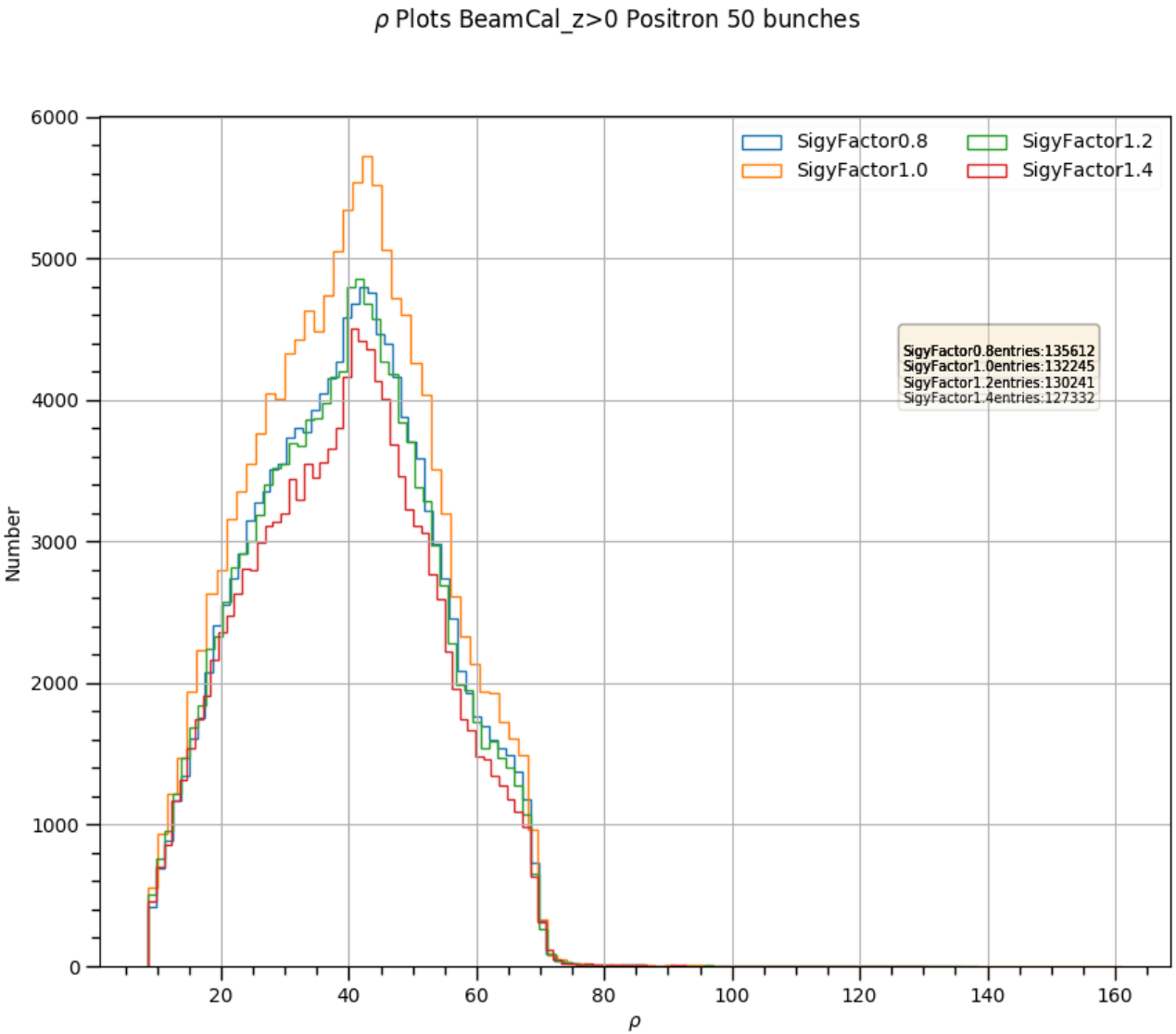
Set 2: ϕ plots for constant σ_x (factor 1.0)



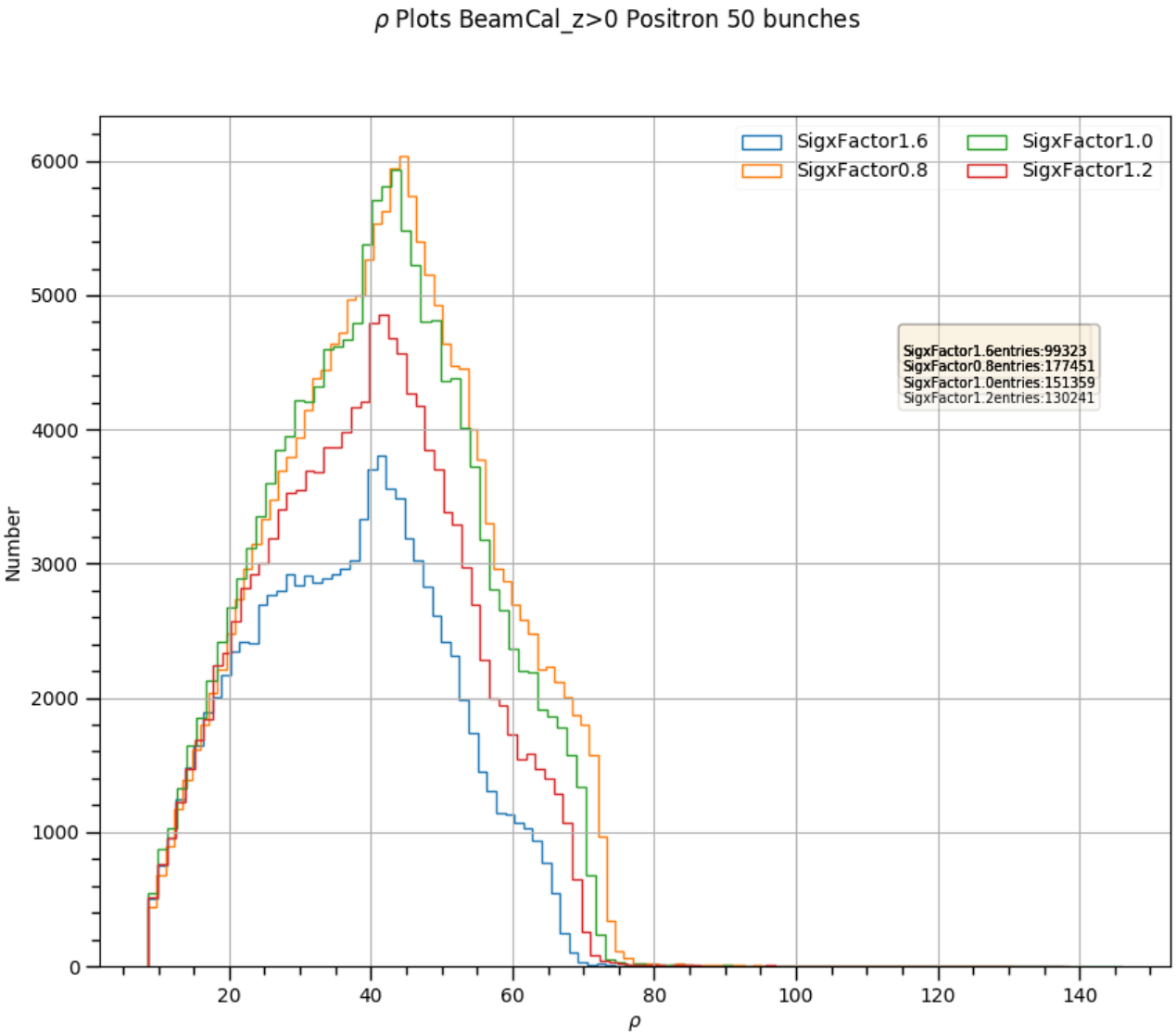
Set 2: ϕ plots for constant σ_y (factor 1.0)



Set 2: ρ plots for constant σ_x (factor 1.0)



Set 2: ρ plots for constant σ_y (factor 1.0)



CNN Architecture

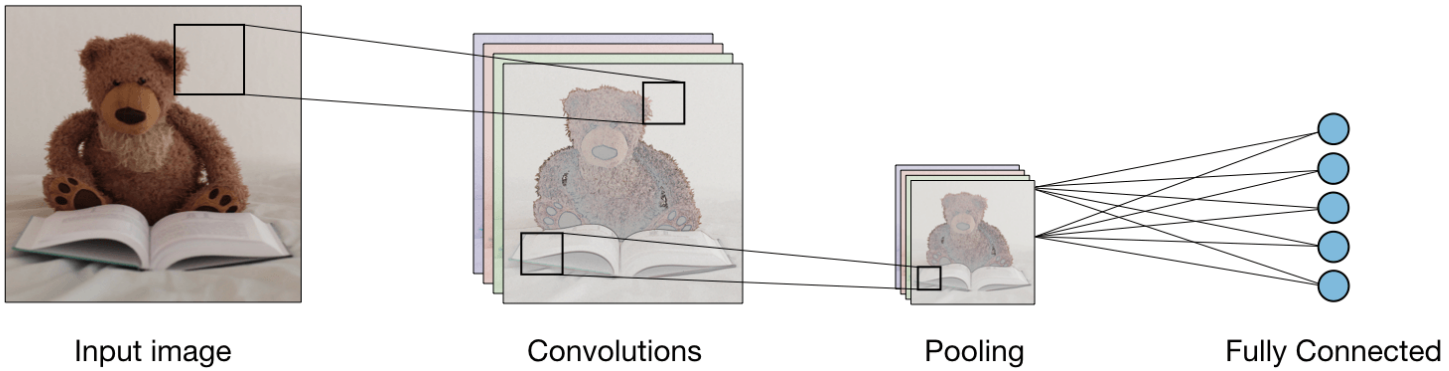


Figure 4: CNN architecture[2]

Types of layers:

- Conv Layer: uses filters that perform convolution operations as it is scanning the input I with respect to its dimensions. Hyper parameters: Filter size F and Stride S
- Max Pooling: Maximum value within a filter is collected
- Fully connvected (FC): operates on a flattened input where each input is connected to all neurons

CNN in two minutes

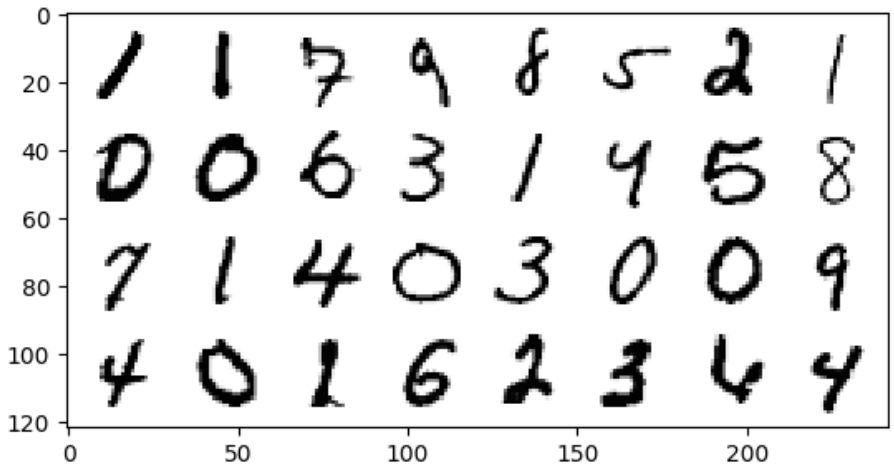
- Activation Function: RelU (Rectified Linear Unit), defined by $g(z) = max(0, z)$ non-linearity has been applied after every Conv and FC layers.
- Softmax: takes as input a vector of scores $x \in \mathbb{R}^n$ and outputs a vector of output probability $p \in \mathbb{R}^n$ $p = (p_1 \dots p_n)$ where $p_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$
- Softmax Cross Entropy: $CE = -\sum_i^C t_i log(f(s_i))$ where t_i is the true label and $f(s_i)$ is the softmax output. For two classes: $CE = -\sum_{i=1}^{C=2} t_i log(f(s_i)) = -t_1 log(f(s_1)) - (1 - t_1) log(1 - f(s_1))$
- Each sample can belong to ONE of C classes. The CNN will have C output neurons that can be gathered in a vector s (Scores). The target (ground truth) vector t will be a one-hot vector with a positive class and $C - 1$ negative classes.

machine Learning Frameworks

- PyTorch is a deep learning framework designed by Facebook AI
- CNN has been carried out using pytorch
- Some tools from SkLearn has been used for visualization and data division

Testing by MNIST data

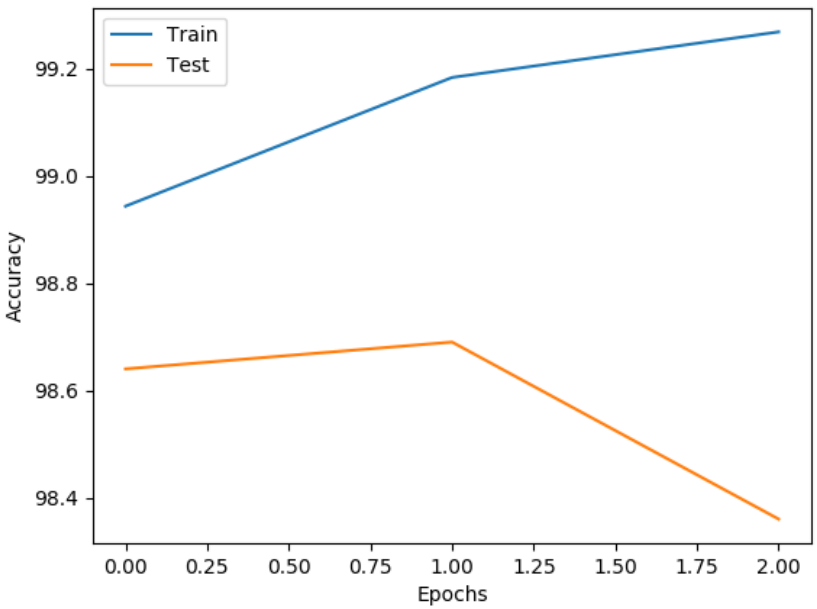
- MNIST data are hadwritten digits
  and have 10 labels
- They are used as standards to make sure code is running properly

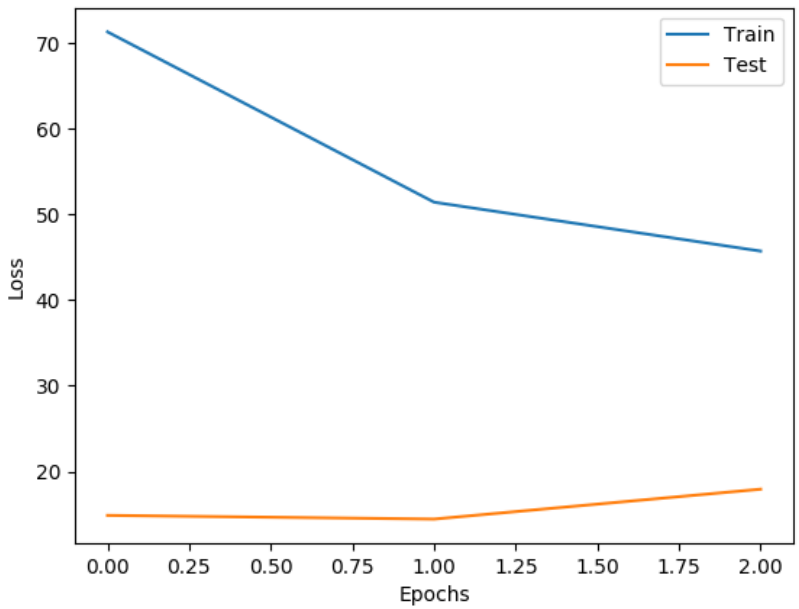


Architecture used

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 28, 28]             640
              ReLU-2         [-1, 64, 28, 28]               0
            Conv2d-3         [-1, 64, 28, 28]          36,928
              ReLU-4         [-1, 64, 28, 28]               0
         MaxPool2d-5         [-1, 64, 14, 14]               0
            Linear-6               [-1, 1024]      12,846,080
              ReLU-7               [-1, 1024]               0
            Linear-8                [-1, 512]         524,800
              ReLU-9                [-1, 512]               0
           Linear-10                 [-1, 10]           5,130
================================================================
Total params: 13,413,578
Trainable params: 13,413,578
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 1.65
Params size (MB): 51.17
Estimated Total Size (MB): 52.82
----------------------------------------------------------------
```

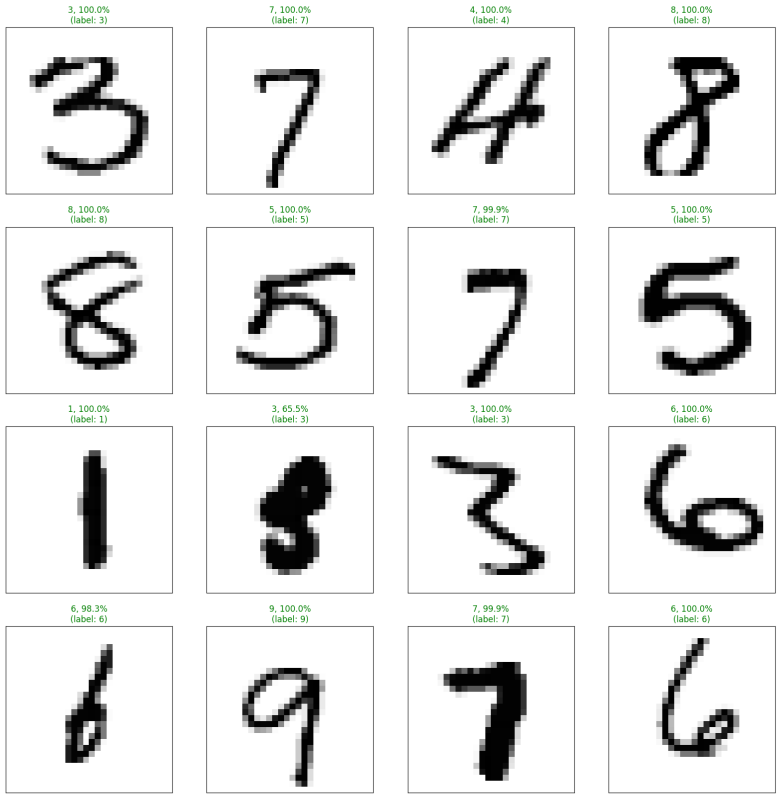
Loss Accuracy



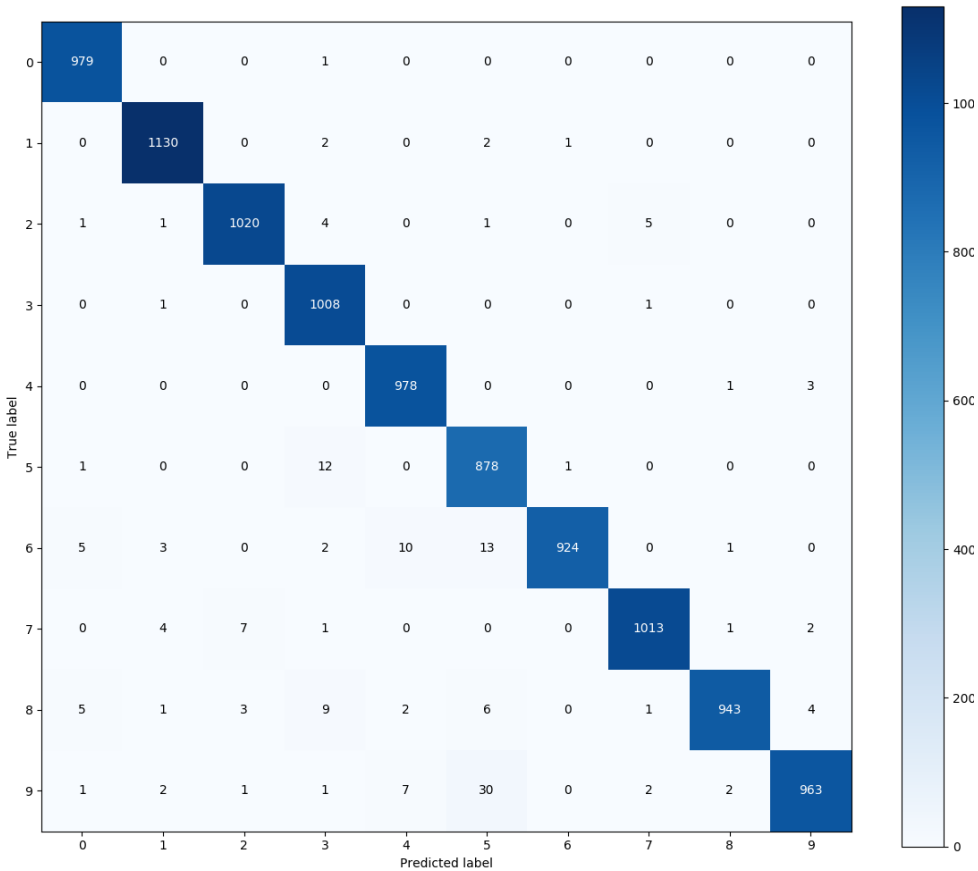


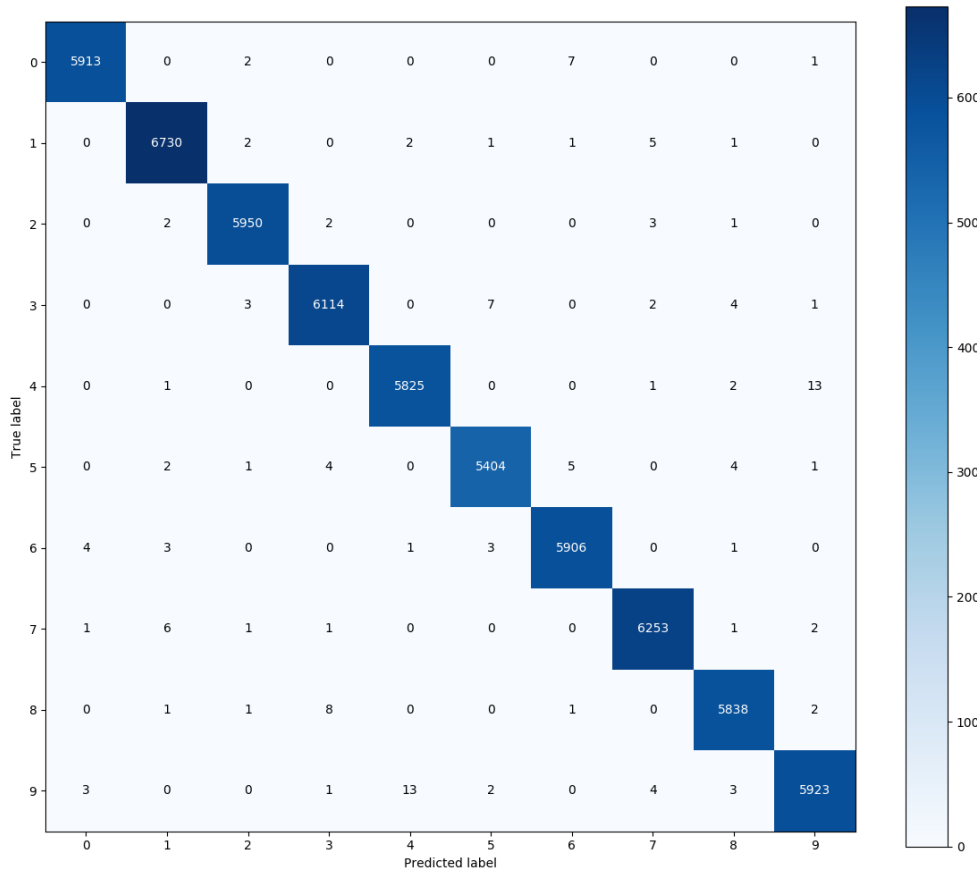
Viewing predictions on random batches

- MNIST data are hadwritten digits and have 10 labels
- They are used as standards to make sure code is running properly
- The figure shows the labeled and predicted images from random batches



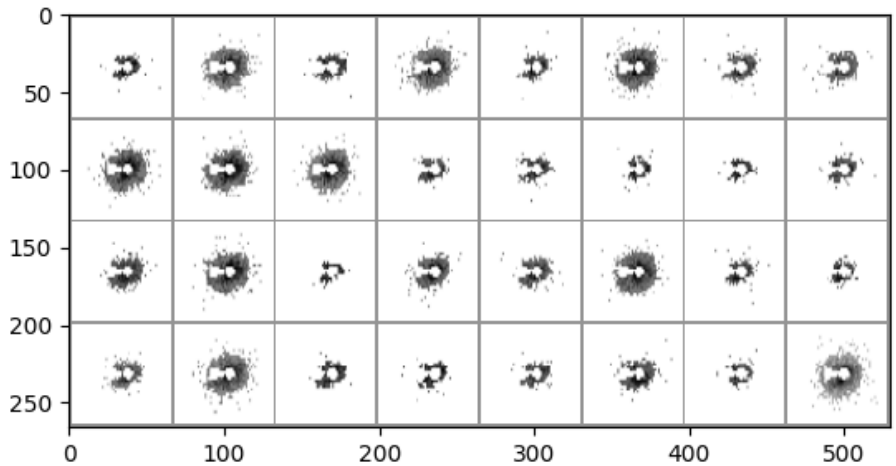
Confusion Matrix: Test and Train





Testing by Set1 data

- Several images from different classes
  are shown in grid
- Set 1 data which contains 16 labels
  have strong signals

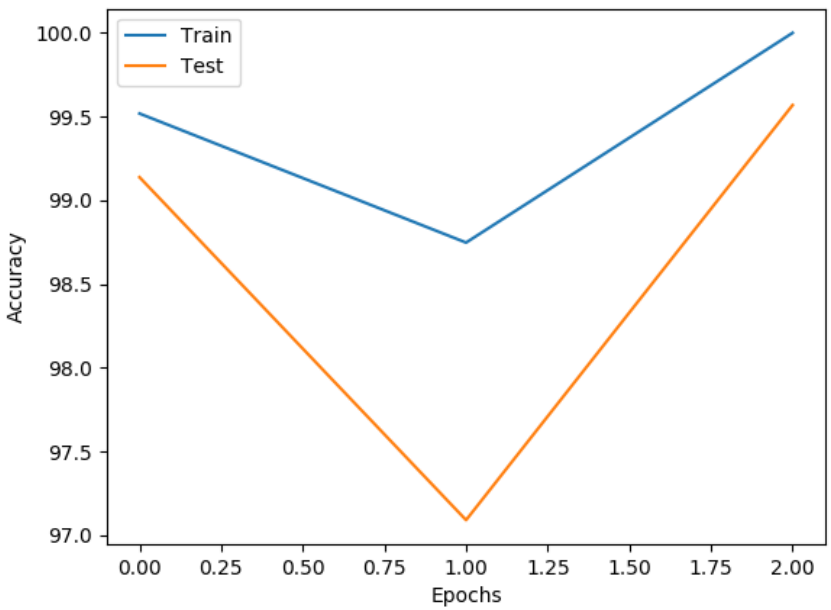


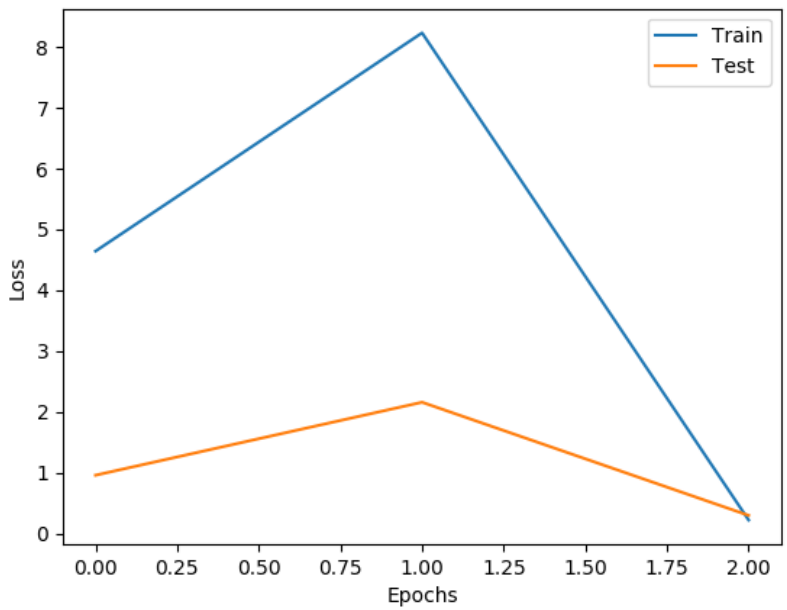
Architecture used

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 64, 64]             640
              ReLU-2         [-1, 64, 64, 64]               0
            Conv2d-3         [-1, 64, 64, 64]          36,928
              ReLU-4         [-1, 64, 64, 64]               0
         MaxPool2d-5         [-1, 64, 32, 32]               0
            Linear-6               [-1, 1024]      67,109,888
              ReLU-7               [-1, 1024]               0
            Linear-8                [-1, 512]         524,800
              ReLU-9                [-1, 512]               0
           Linear-10                 [-1, 16]           8,208
================================================================
Total params: 67,680,464
Trainable params: 67,680,464
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 8.52
Params size (MB): 258.18
Estimated Total Size (MB): 266.72
----------------------------------------------------------------
```

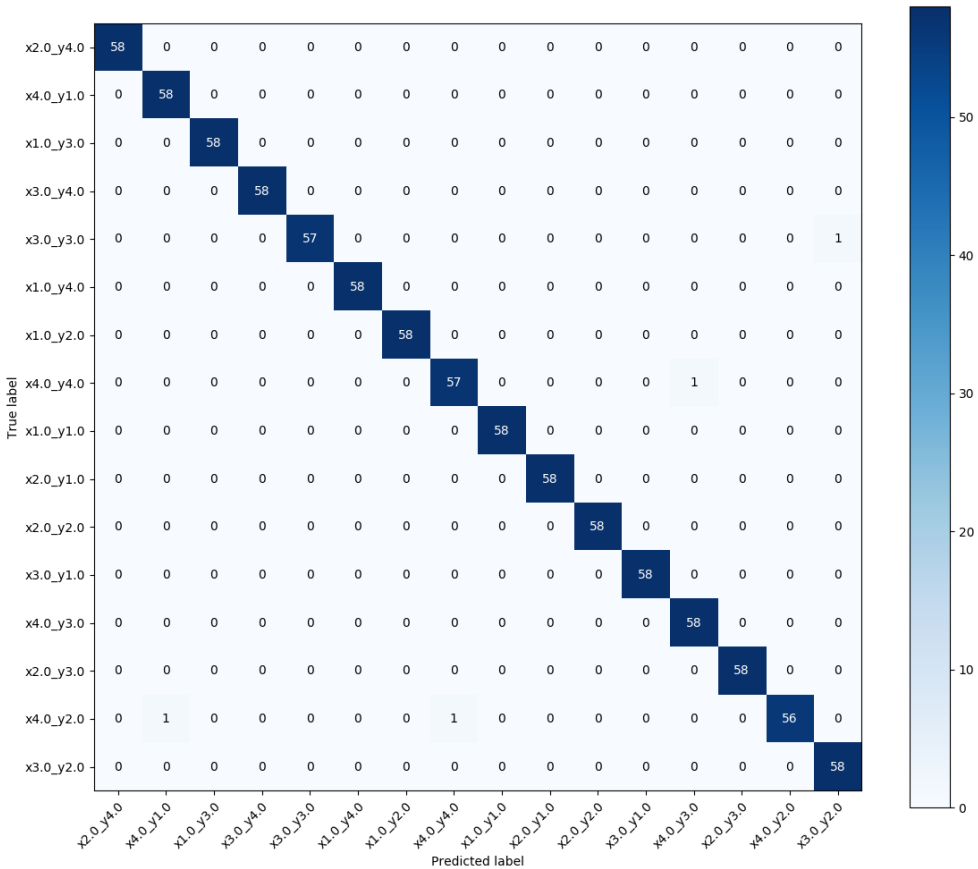
Loss Accuracy

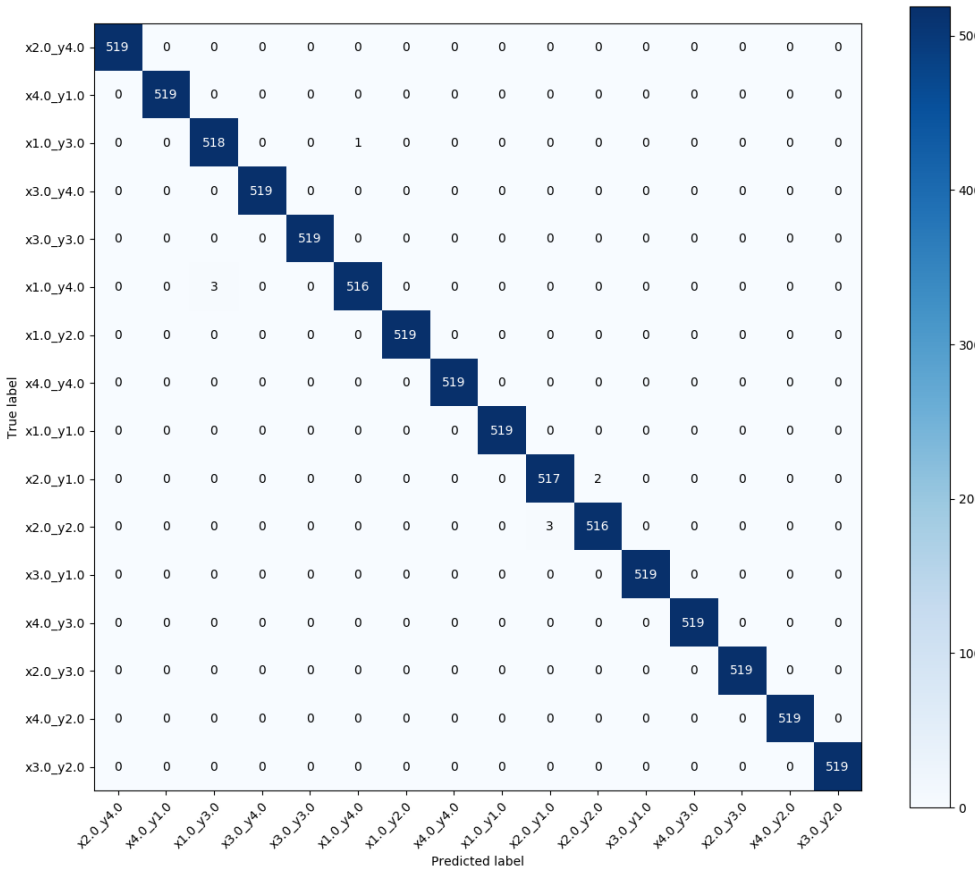
- Set 1 data have very strong signals
- High accuracy in just 2 epochs





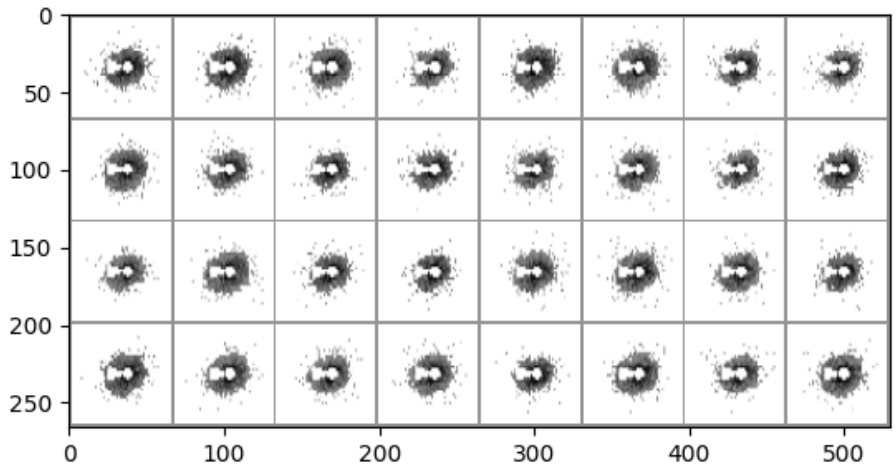
Confusion Matrix: Test and Train





Set 2 data

- The images are not very discriminatory
- Signals are not as strong
- Batch Normalization has been included after every conv layer

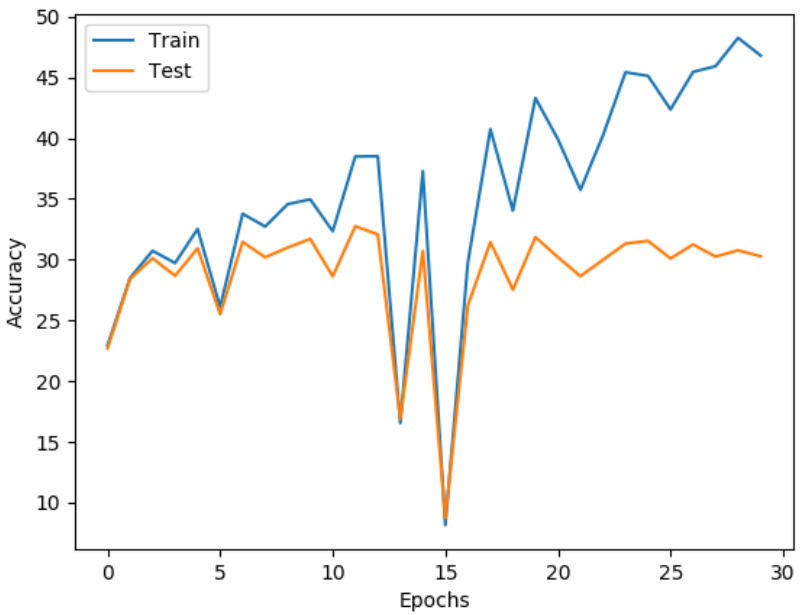


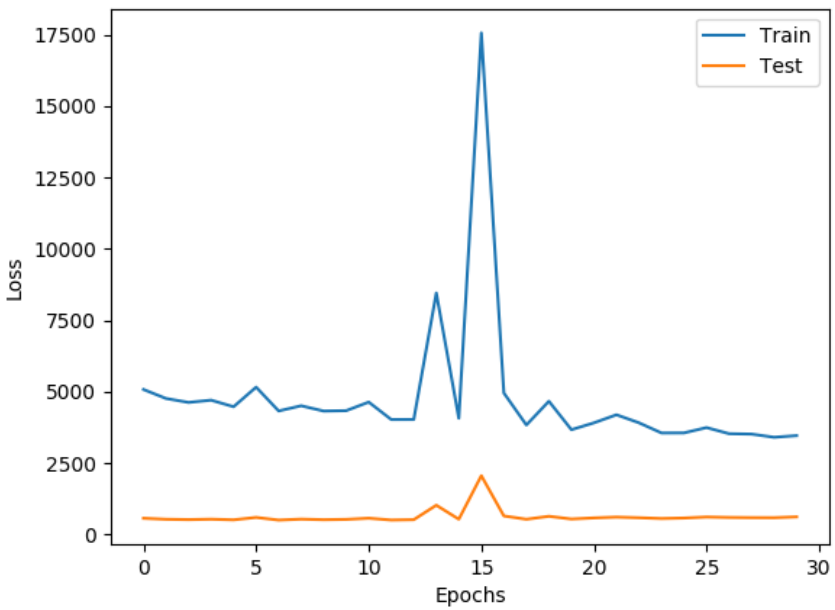
Architecture used

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 64, 64]             640
       BatchNorm2d-2         [-1, 64, 64, 64]             128
              ReLU-3         [-1, 64, 64, 64]               0
            Conv2d-4         [-1, 64, 64, 64]          36,928
       BatchNorm2d-5         [-1, 64, 64, 64]             128
              ReLU-6         [-1, 64, 64, 64]               0
         MaxPool2d-7         [-1, 64, 32, 32]               0
            Linear-8               [-1, 1024]      67,109,888
              ReLU-9               [-1, 1024]               0
           Linear-10                [-1, 512]         524,800
             ReLU-11                [-1, 512]               0
           Linear-12                 [-1, 25]          12,825
================================================================
Total params: 67,685,337
Trainable params: 67,685,337
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.02
Forward/backward pass size (MB): 12.52
Params size (MB): 258.20
Estimated Total Size (MB): 270.74
----------------------------------------------------------------
```

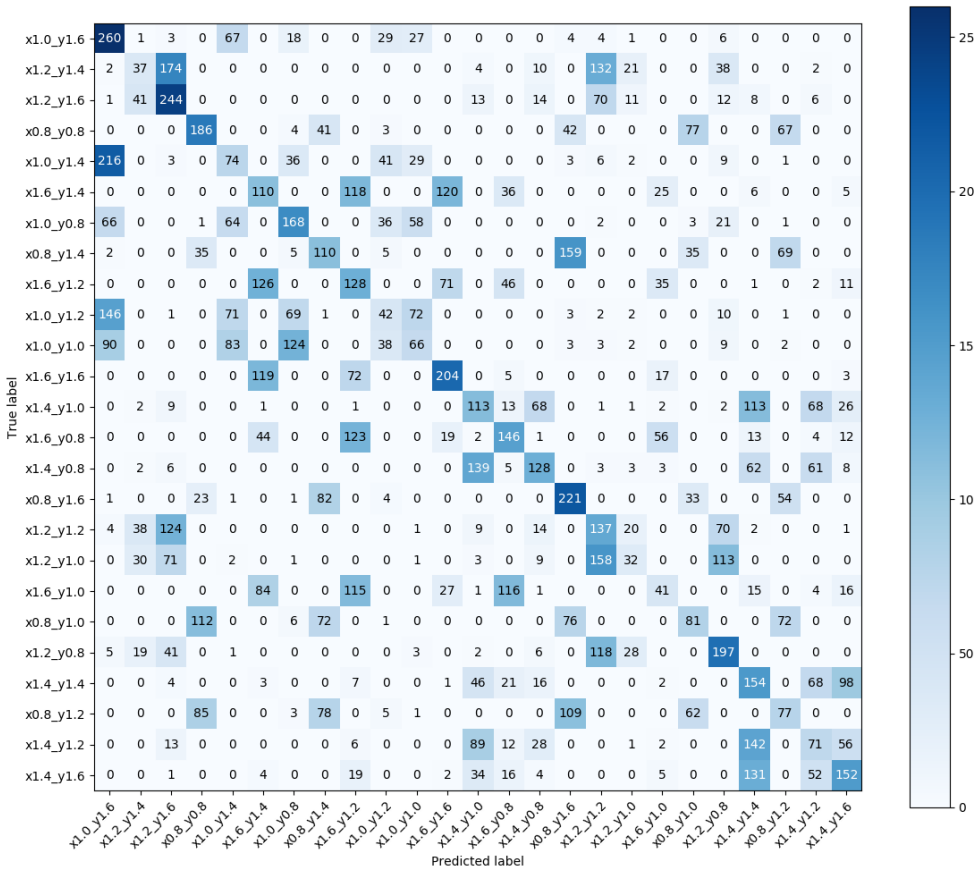
Loss Accuracy

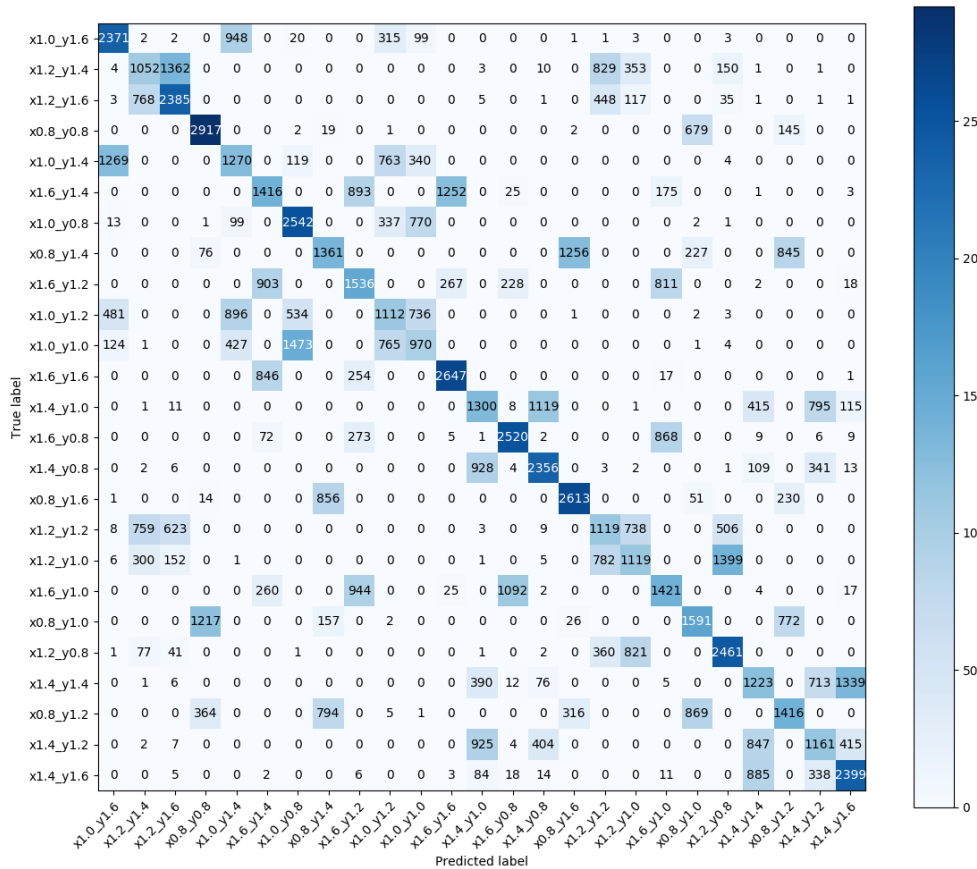
- Loss displayed are running losses; Should have plotted loss per dataset length
- Overfitting occurs after 15 epochs
- The one entry hits with high ρ values might be a reason because they increase sample noise: proper augmentation method would be necessary





Confusion Matrix: Test and Train





Viewing predictions on random batches

Number of entries vs ρ_{mean}

- Differnet colors have been used to visualize different classes
- It will be easier to classify by means of simple machine learning algorithms such as SVM or multilayer perceptron

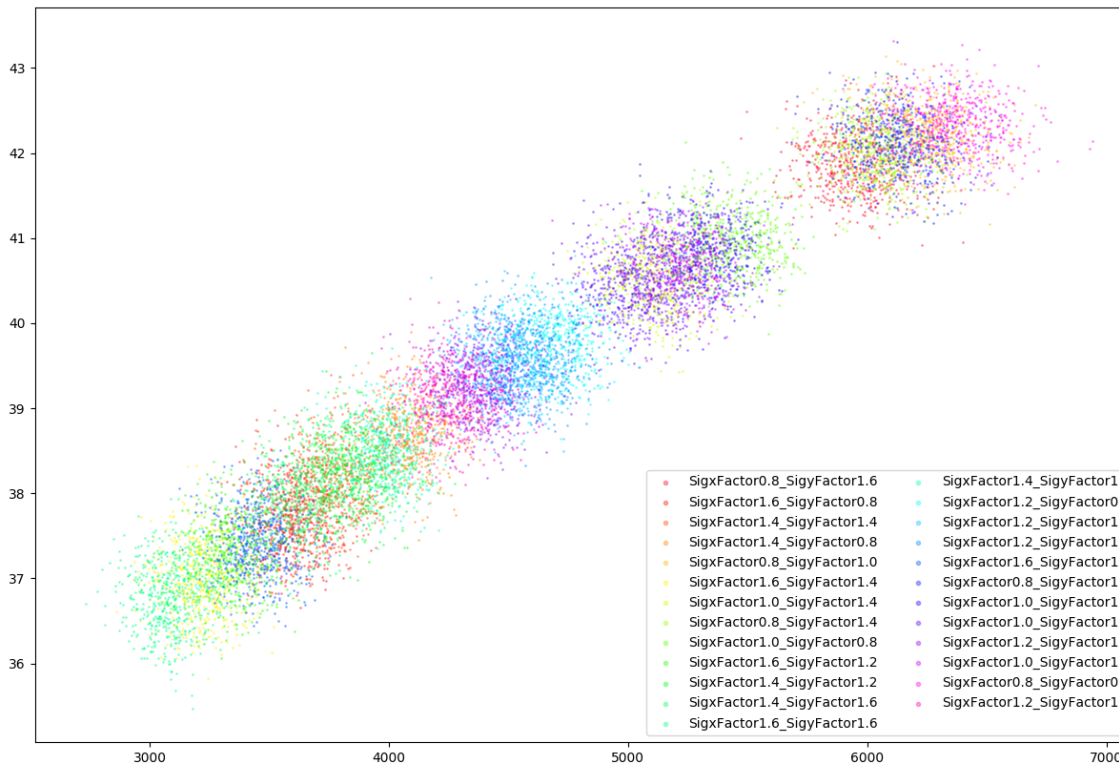


Figure 5: Number of entries: horizontal vs ρ_{mean} vertical

The End