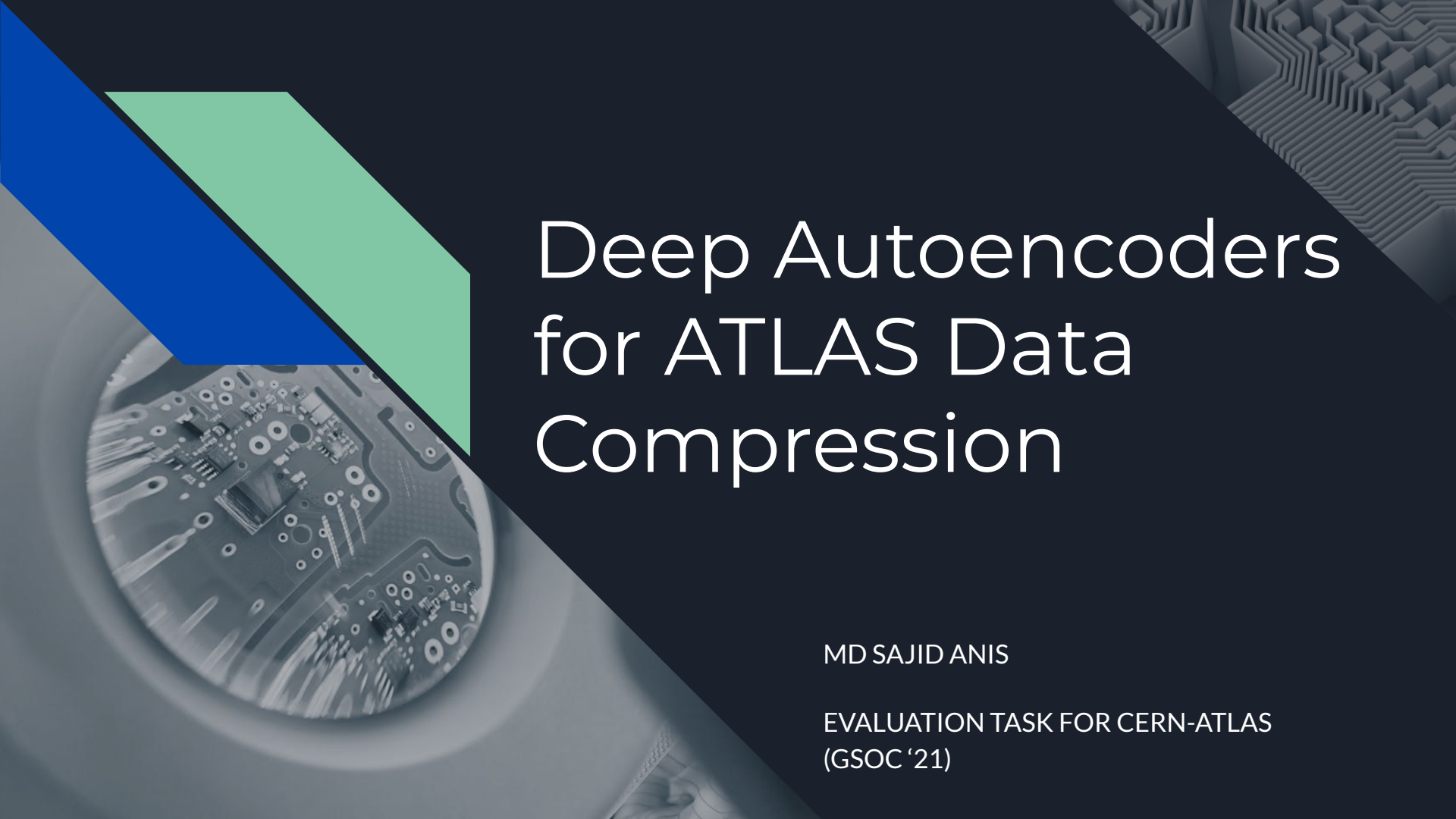


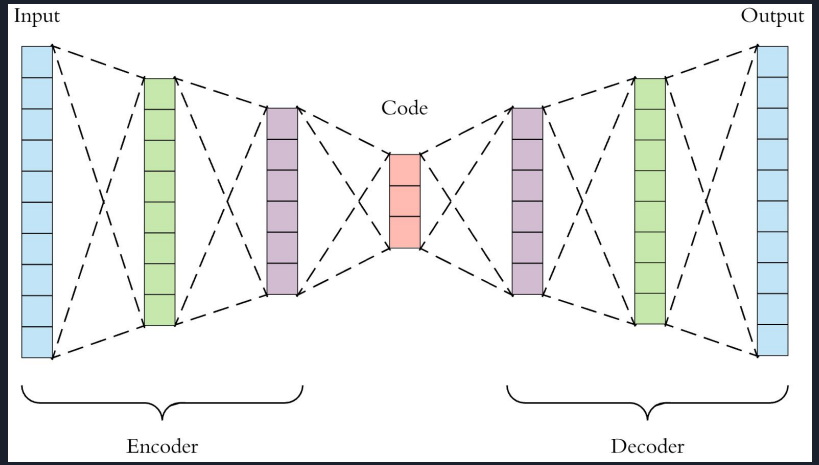
Deep Autoencoders for ATLAS Data Compression

MD SAJID ANIS

EVALUATION TASK FOR CERN-ATLAS
(GSOC ‘21)

DEEP AUTOENCODERS

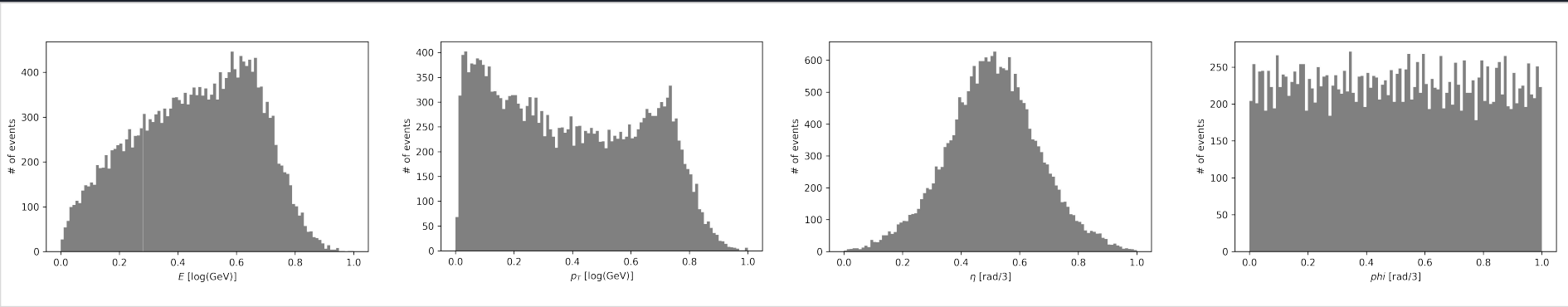
- An Artificial Neural Network used to learn efficient data codings in an unsupervised manner.
- It includes an encoder that transforms the input into lower dimensional representation and a decoder which tries to reconstruct the original data from the lower dimensional representation
- It provides lossy compression for the original input, though it can learn the non-linear representation of input. That's where it shines.



EVALUATION TASK - Normalizing the Data

The major part of training the neural networks are analysing and preprocessing the data available to us. As we need to compress only the `jet particles`, we need to extract them from the event data. We also need to normalize the data. We will be using custom normalization along with the standard min max normalization of the data with the parameters, ‘E’, ‘pt’, ‘eta’, ‘phi’.

E → (log(E) - min) / (max - min)
Pt → (log(pt) - min) / (max - min)
Eta → (eta - min) / (max - min)
Phi → (phi - min) / (max - min)



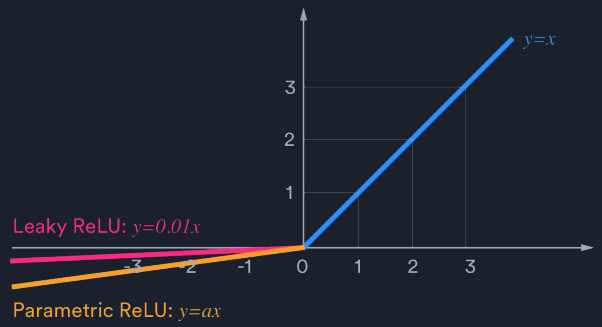
Loss Function and Model Details

As output are symmetrically close to the input in Autoencoders, MSE(Mean Squared Error) would be the preferred loss function.

$$\mathrm{MSE} = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{P}(y_i(\vec{x}_n) - x_{ni})^2 \ ,$$

Model Description:

For the Autoencoder, The model contains 7 fully connected layers with 200, 200, 20, 3, 20, 200, 200 nodes, and LeakyReLU activation function at each layer.



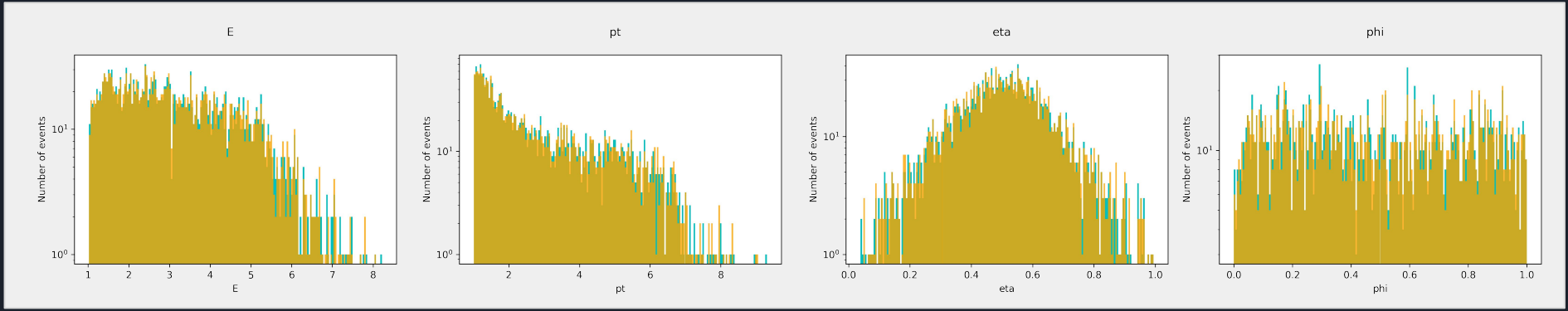
Training and Inference

The data is divided into test set and training set and the training set are further divided into validation and train set. On the train set, we get mse of:

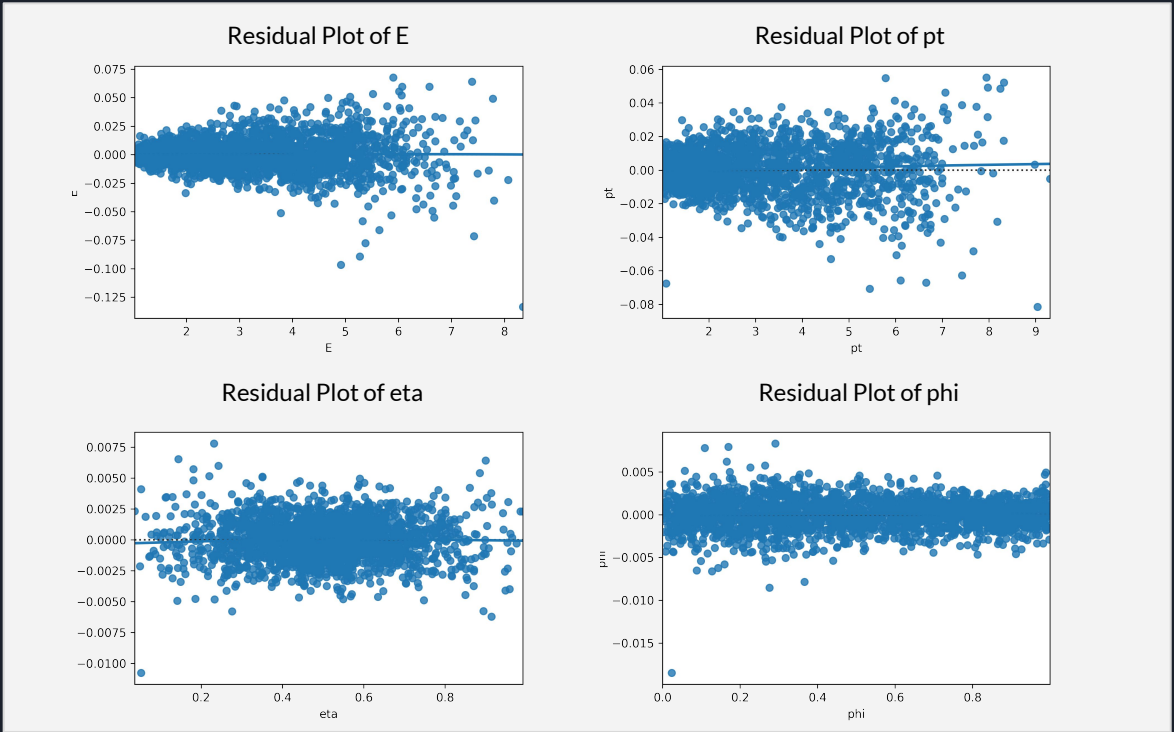
Normalization type	MSE on the train-set	MSE on the validation set
Custom + Standard	5.785044322692556e-06	6.434316674130969e-06

Inference Plots for the Test Data along with Predicted Output

Here, Blue bins are Input Data & Orange bins are Predicted Output from the AutoEncoder. As we can see the predictions are almost matching the waveform of our input data.



Residual Plots of our model on all Dimension



As we can see this model with the custom normalization, provides better performance with little variance for most of the parameters.

An example showing the working of encoder and decoder separately

```
[18]: e_code = model.encode(torch.tensor([0.527866 ,0.043150 ,0.113446 ,0.190413]))

[19]: # Here we can see that our model actually reduce the dimension from 4-D to 3-D
      e_code

[19]: tensor([ 0.3362,  0.9150, -4.7212], grad_fn=<AddBackward0>)

[20]: output = model.decode(e_code)

[21]: # And here it recreates the 4-D data from our encoded lower dimensional input
      output

[21]: tensor([0.5835, 0.0229, 0.1215, 0.1918], grad_fn=<AddBackward0>)
```