Data Compression for Jets

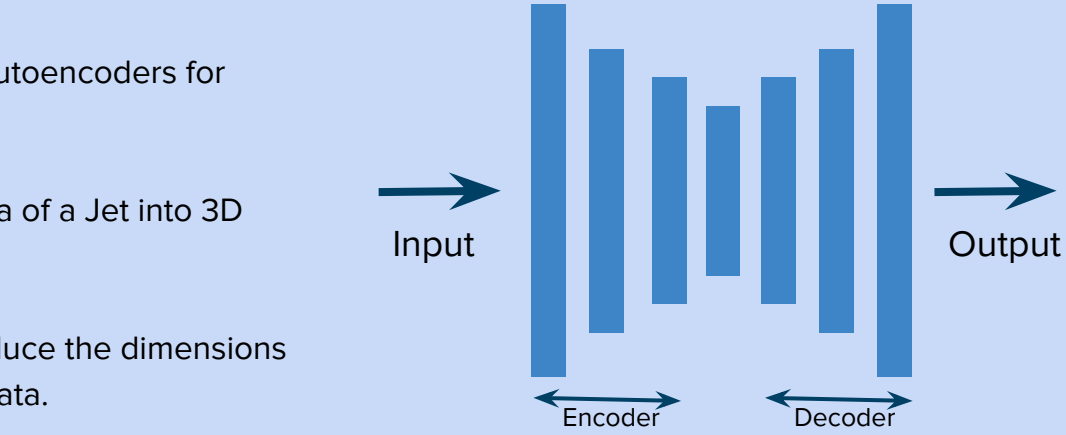
Aniket Prasad
National Institute of Technology
Agartala

Problem in the Hand

- We are provided with the data from the experiments to detect different objects. Consisting of recorded trigger instances which has been cleaned and selected accordingly.
- We have to reduce the 4 essential data of an object to 3 dimensions which makes it a problem where Neural Networks can help.

Deep-compression of Jets

- Deep compression refers to usage of autoencoders for performing data compression.
- Here we basically compress the 4D data of a Jet into 3D equivalents.
- The idea is to use Deep Learning to reduce the dimensions of data without any noticeable loss of data.
- At the same time, the compression is automated and thus evidently faster.



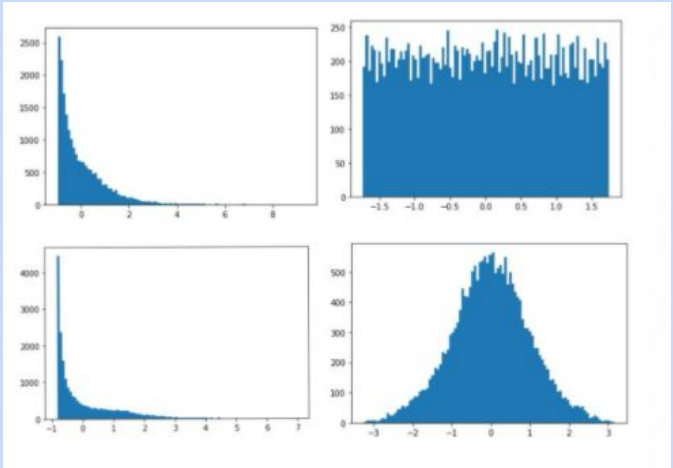
A typical autoencoder (encoder+decoder) network

Data Processing

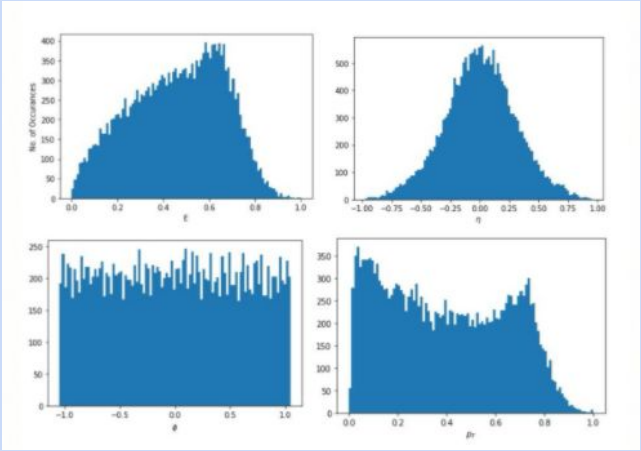
- The Data is distributed with different ranges of values. We ensure model is not affected by unequal weights of different columns(which are essentially dimensions here) when we go from 4D to 3D.

Application of Normalisation

- After experimentation to reduce the unbalanced weightage of each column, I settled on using the **custom normalisation parameters** mentioned by Eric Wulff in his master's thesis.
- Also applied **MinMaxScalar** (basically to further scale it down to [0,1] using predefined methods of python.



Standard normalization



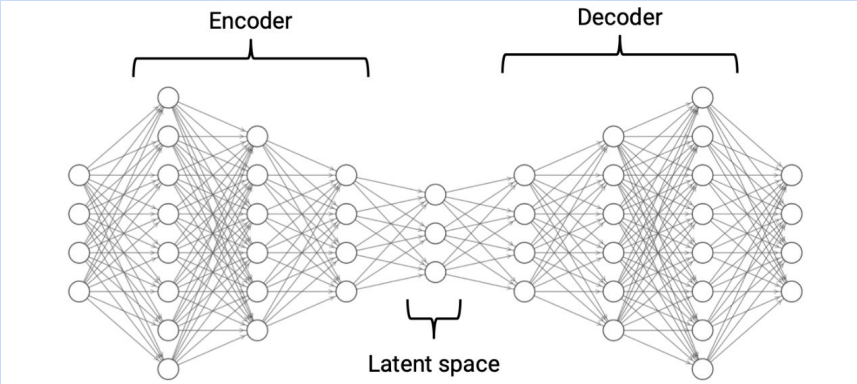
Custom normalization

Model Training and Learning

- Ideally, the decompressed 4D data should be same as the input data and hence we train the autoencoder by using a mean square error loss as the optimization loss function.

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

- I performed training using a Deep Learning model of 7 layers Custom Normalised Data.
- SInce we need to detect patterns in very less number of layers and ensure nothing is missed, I use **non-linear activation function called Tanh activation function.**
- To make learning more efficient without overfitting, we use **Learning Rate Scheduling** techniques here.
- Also to ensure Encoder can be more generalised, we apply **weight decay**(***=1e-6***) after each cycle.



Model description:

The model contains **7 fully-connected layers with 200, 100, 50, 3, 50, 100, 200 nodes and Tanh activation laye**r after each layer.

Results

I have plotted the reconstructed data along with the provided data. Vast Losses will be clearly visible if that occurs.

In my case they overlap farely well for all the 4 data. I also quantitatively tried to find median deviation or the loss.

Observations:

- The validation loss after training for 100 epochs was of the order **1e-6** with no signs of overfitting.
- **As I had taken only the jet objects, this can work pretty well in jet data with these filters as mentioned in the description.**
- **Mean of residuals was 0.04838729640529177**
- **As a further probe, I would like to train this encoder on top of classifier network for generalised data.**