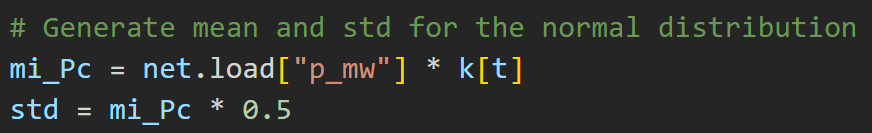
# Neural Network for predicting voltages levels: Report 1

06/09/2022

The last time we spoke I would generate new samples to train the Neural Network. So, I adapted your code of the Monte Carlo Method and changed the standard deviation parameter to create datasets with more variability. In Figure 1 there is part of the code where the mean and standard deviation parameters were defined for later generating random samples from a normal (Gaussian) distribution.

Figure 1 – Code definition of the mean and std parameters



Percentage value to change the std

Loads demands

Loadshape

The loads demands are the standard loads that came with the network, case33bw from Matpower. The loadshape that I used is depicted in Figure 2. Also, the percentages that I used to change the std were 0.05, 0.25, and 0.5.

I generated 1000 samples for each std value, more than that (e.g. 5000) takes too long to process and is not worthy it just for this report. These samples were divided into 70% for training and validation and 30% for testing.

Ps: I didn’t understand why the mean and std were calculated like this and what percentage values were acceptable.

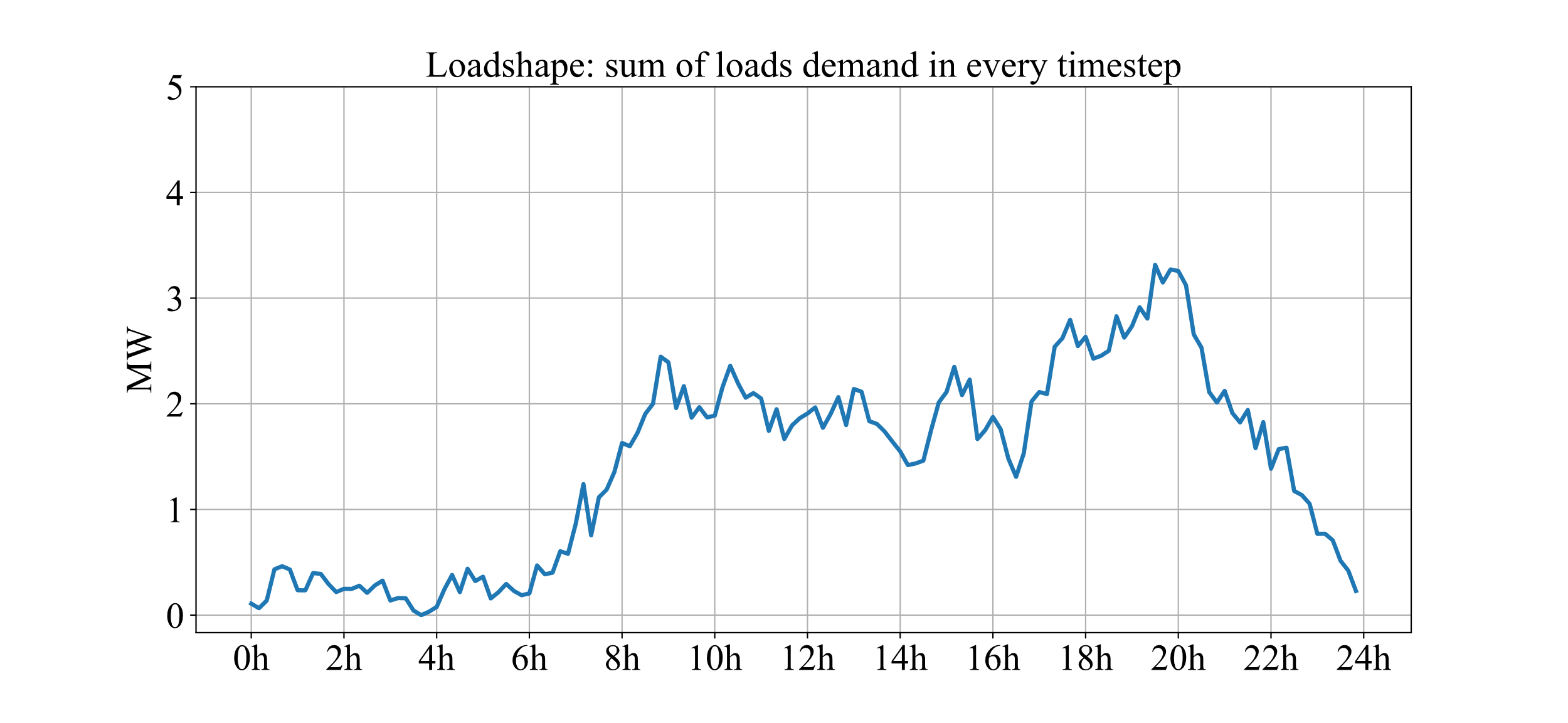


Figure 2 - Loadshape of 144pts used in the code.

The best Neural Network model that I found, from previous tests, was a 3-layer NN with 89 input nodes, 190 hidden nodes, and 40 output nodes (Figure 3). The description of the input and output layers is shown in Table 1. The NN was trained with batch sizes of 150 and 100 epochs, using ReLU activation function and the Adam optimizer.

|  |  |  |
| --- | --- | --- |
| Number of features | Input Layer | Output Layer |
| 2 | Reference Voltages from Transformers 1 and 2 | Voltages levels of all buses |
| 2 | Active Power demand from Transformers 1 and 2 |
| 3 | Active power generated from PVs in buses 12, 18 and 25 (MV) |
| 2 | Active power generated from PVs in buses 29 and 32 (LV) |
| 40 | Average active power load demands from previous days |
| 40 | Average reactive power load demands from previous days |

Table 1 - Neural Network model description.

In Figure 3 the training loss for each std value is shown and we can see how they decrease over time.

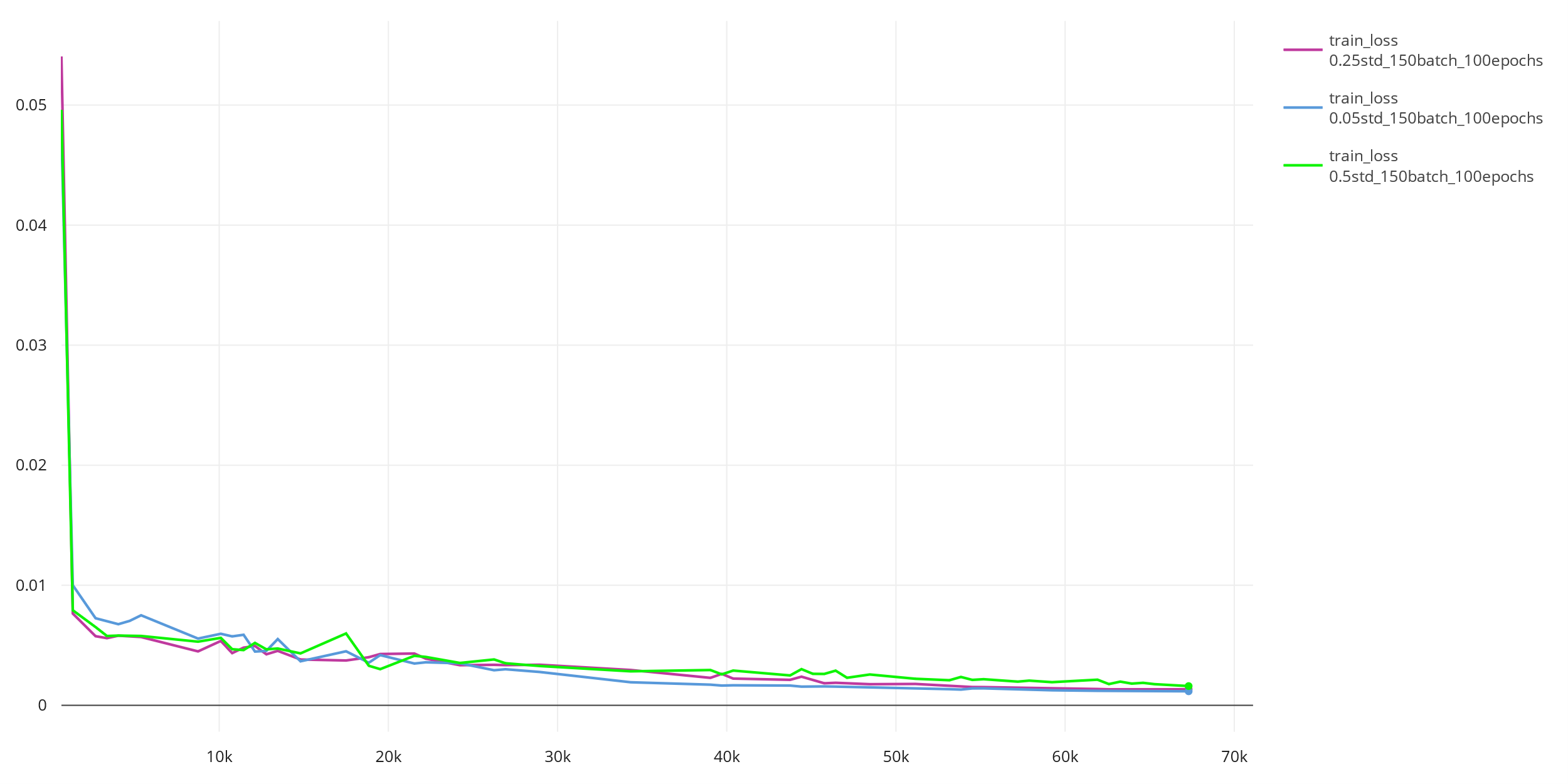


Figure 3 - Train loss over training steps

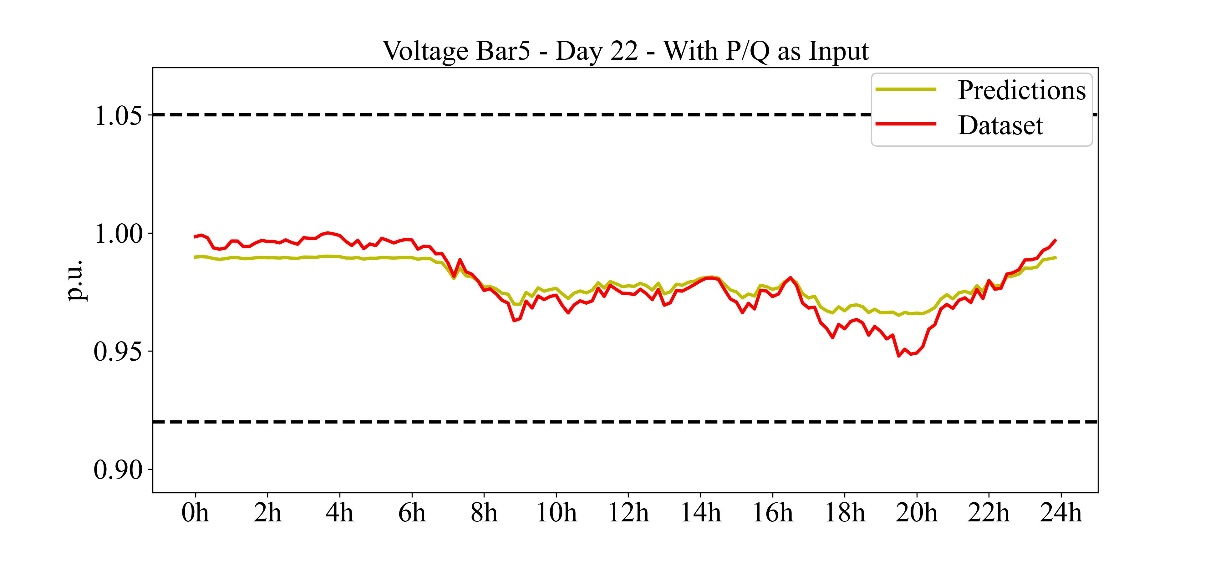
The errors and the voltages levels of different bars (5, 15, 38) with the predicted value and the real dataset values from the test set are shown below.

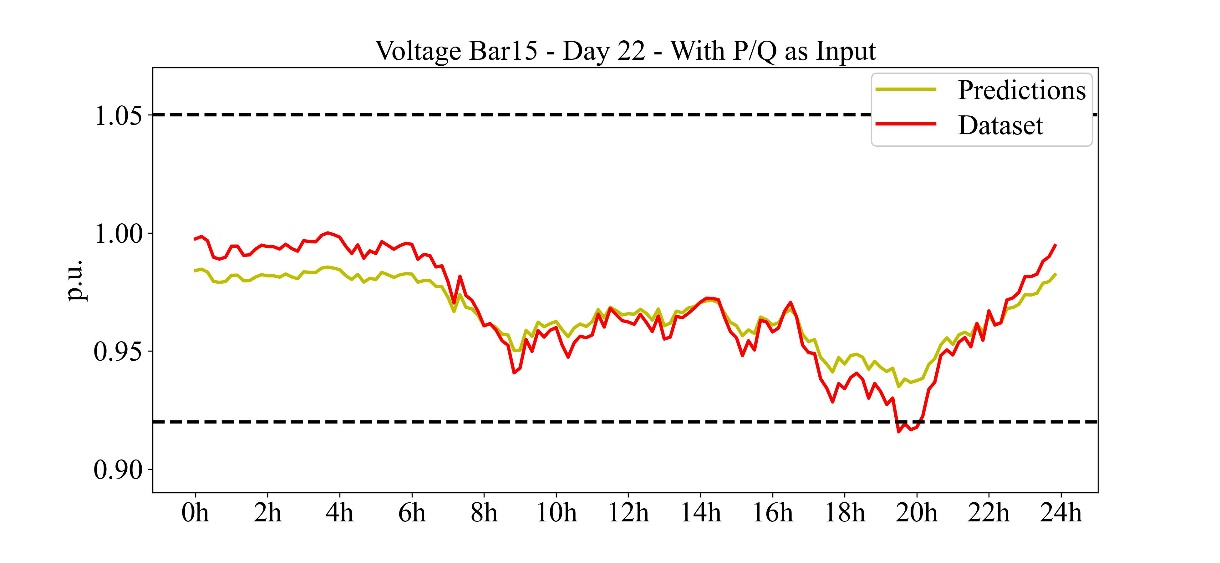
* Std equal to 0.05 of the mean

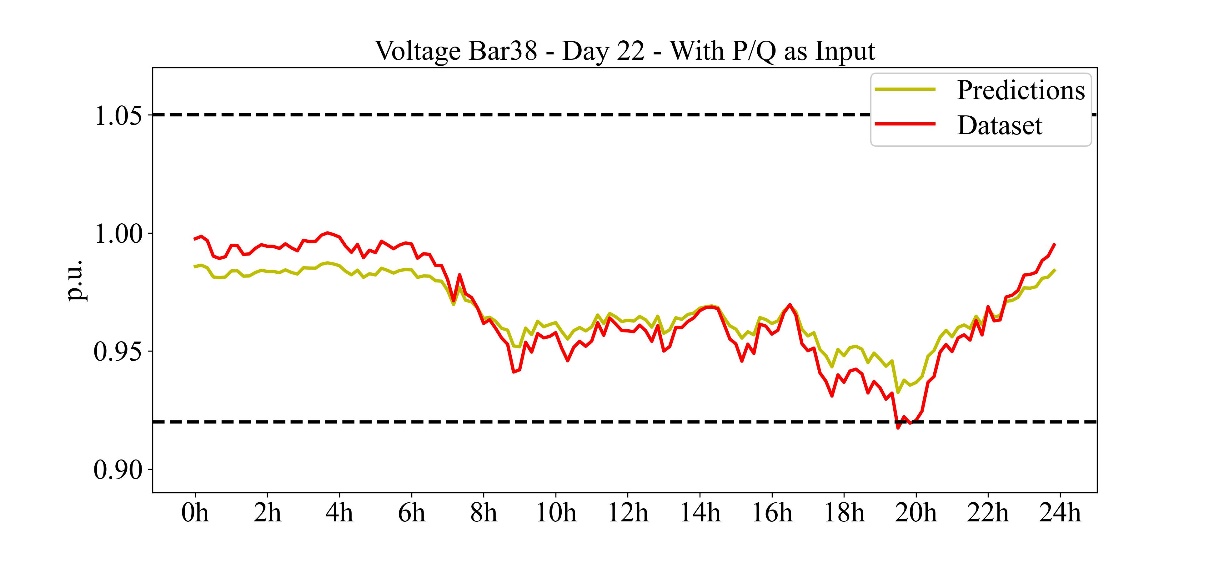
The errors in the test set were:

Mean Absolute Error: 0.00554

Mean Squared Error: 0.00714





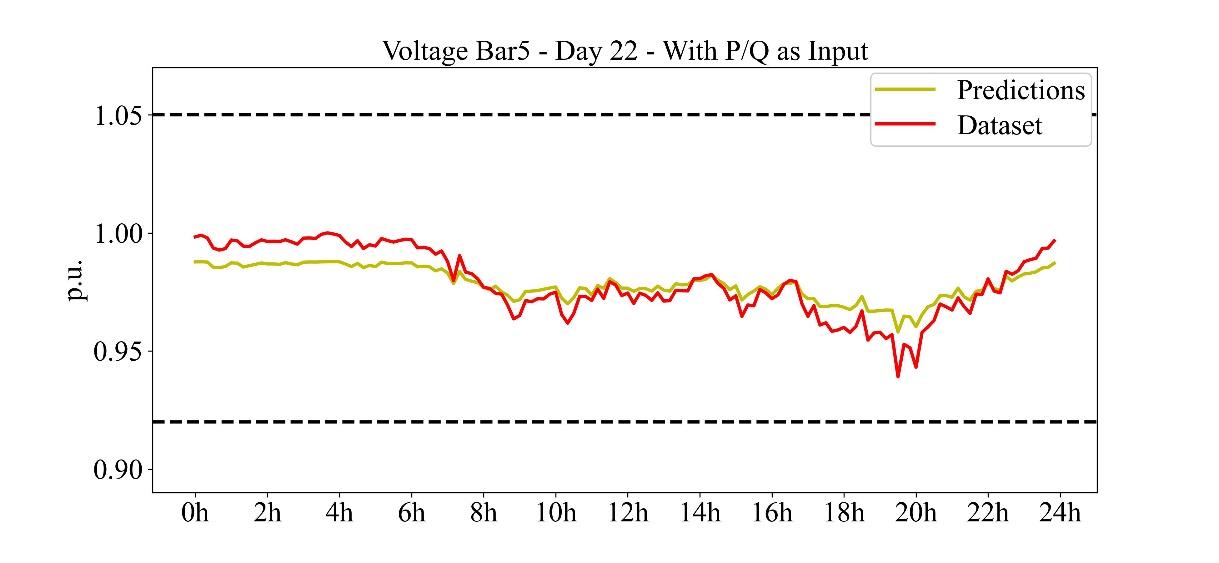


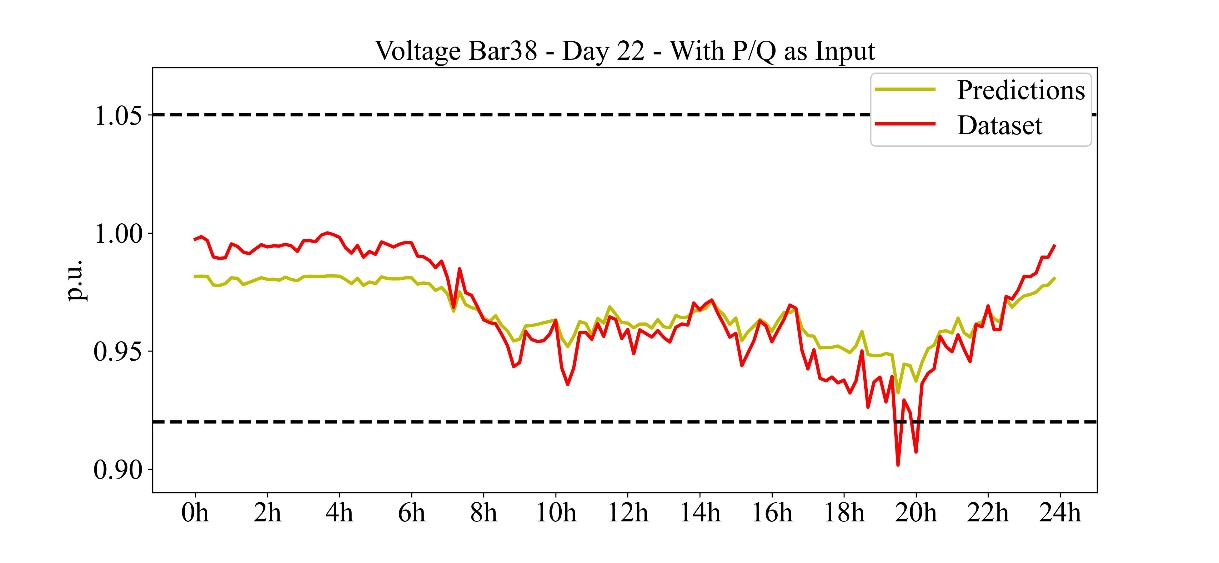
* Std equal to 0.025 of the mean

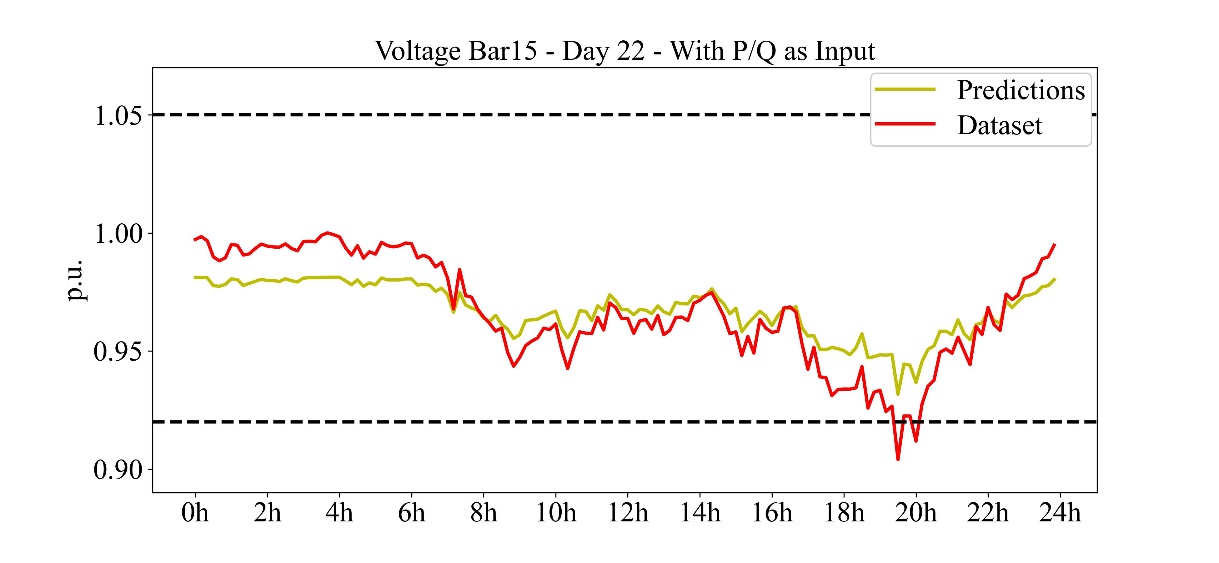
The errors in the test set were:

Mean Absolute Error: 0.00603

Mean Squared Error: 0.00791





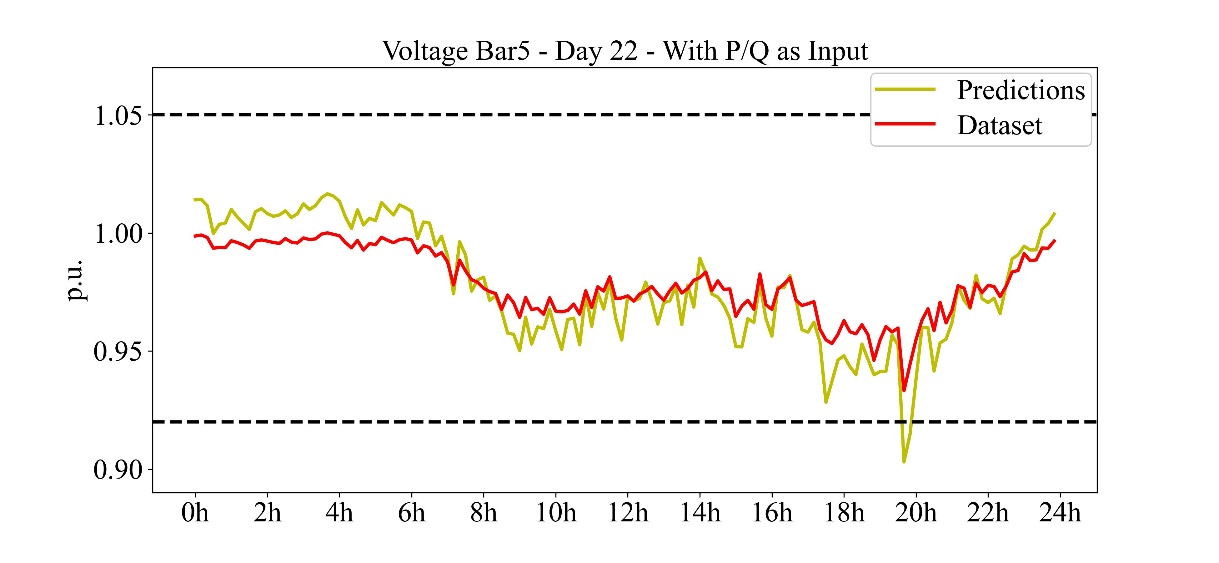


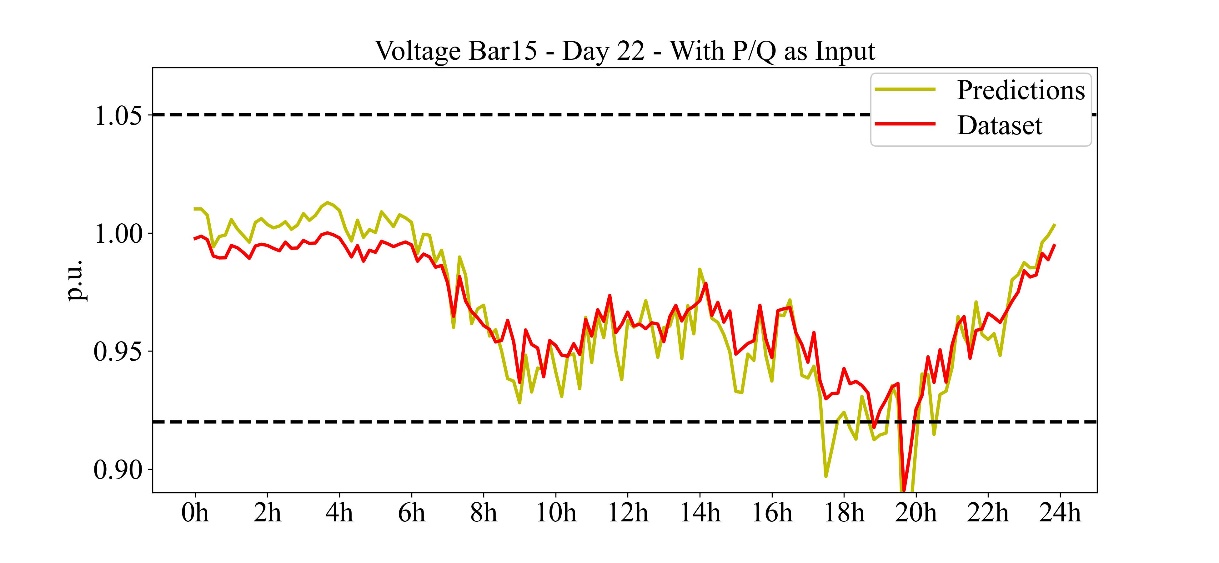
* Std equal to 0.5 of the mean

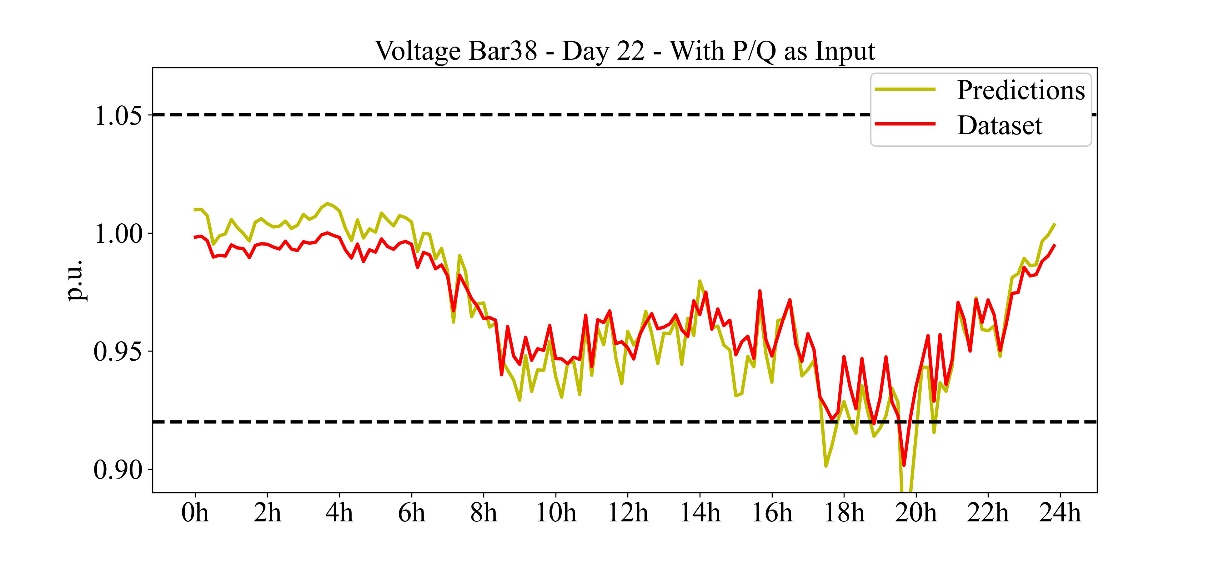
The errors in the test set were:

Mean Absolute Error: 0.00752

Mean Squared Error: 0.00921







In summary, the results are quite good with low errors in the test set. The only problem is when we look at the validation accuracy (the accuracy measured in the validation set while training) (Figure 4), as we can see it variates a lot with time and does not converge for a fixed value.

So, maybe the low errors in the test set are given by the fact that we are working with small values on the output (voltage in p.u.) so it's easier to take a guess, and not because the NN learned to predict it. Testing this neural network with another dataset could be a better metric to see how well it can predict those voltages.

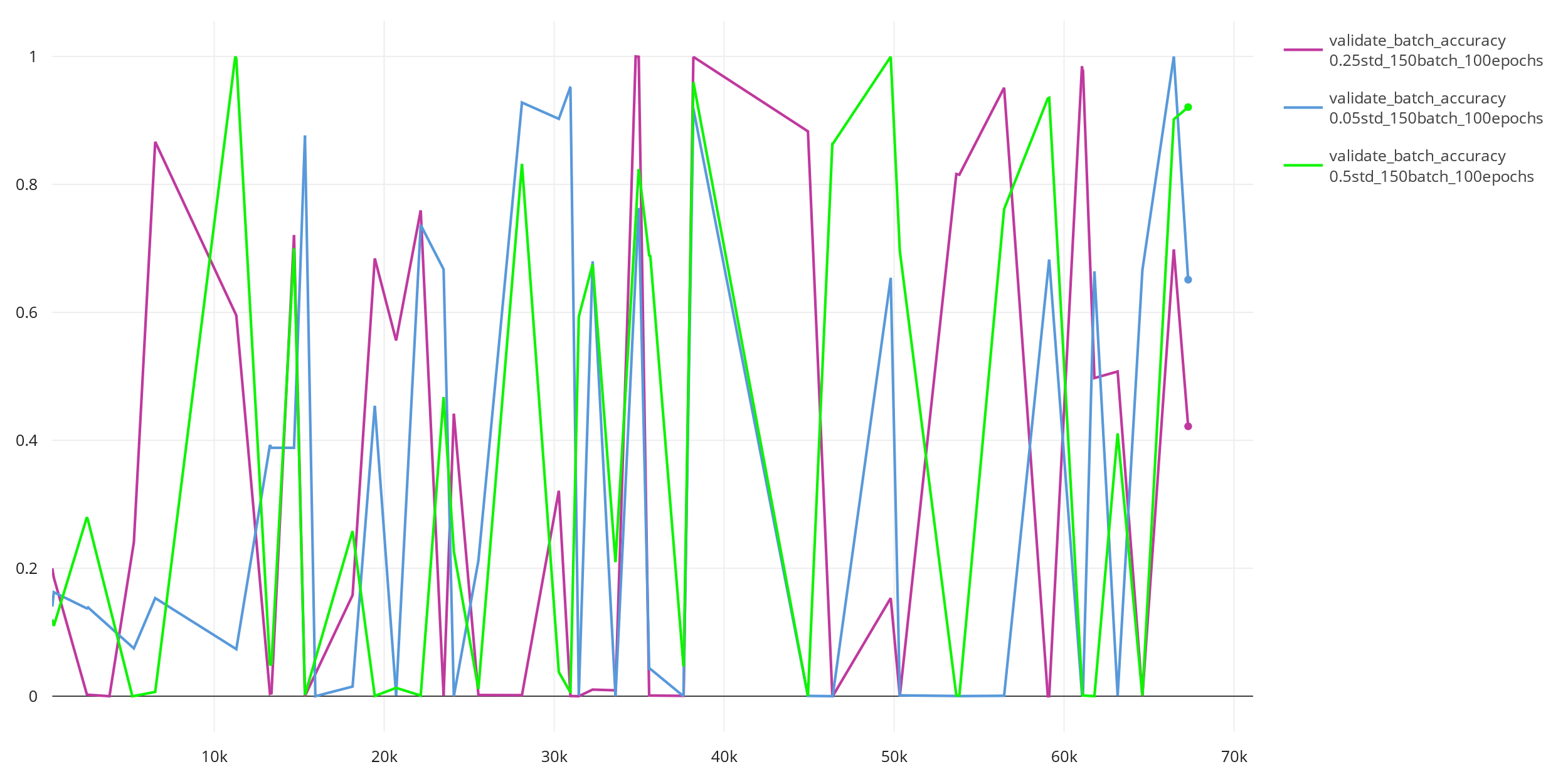


Figure 4 - Validation accuracy over trainig steps