Practical deep learning
Assignment 2

GitHub project: 

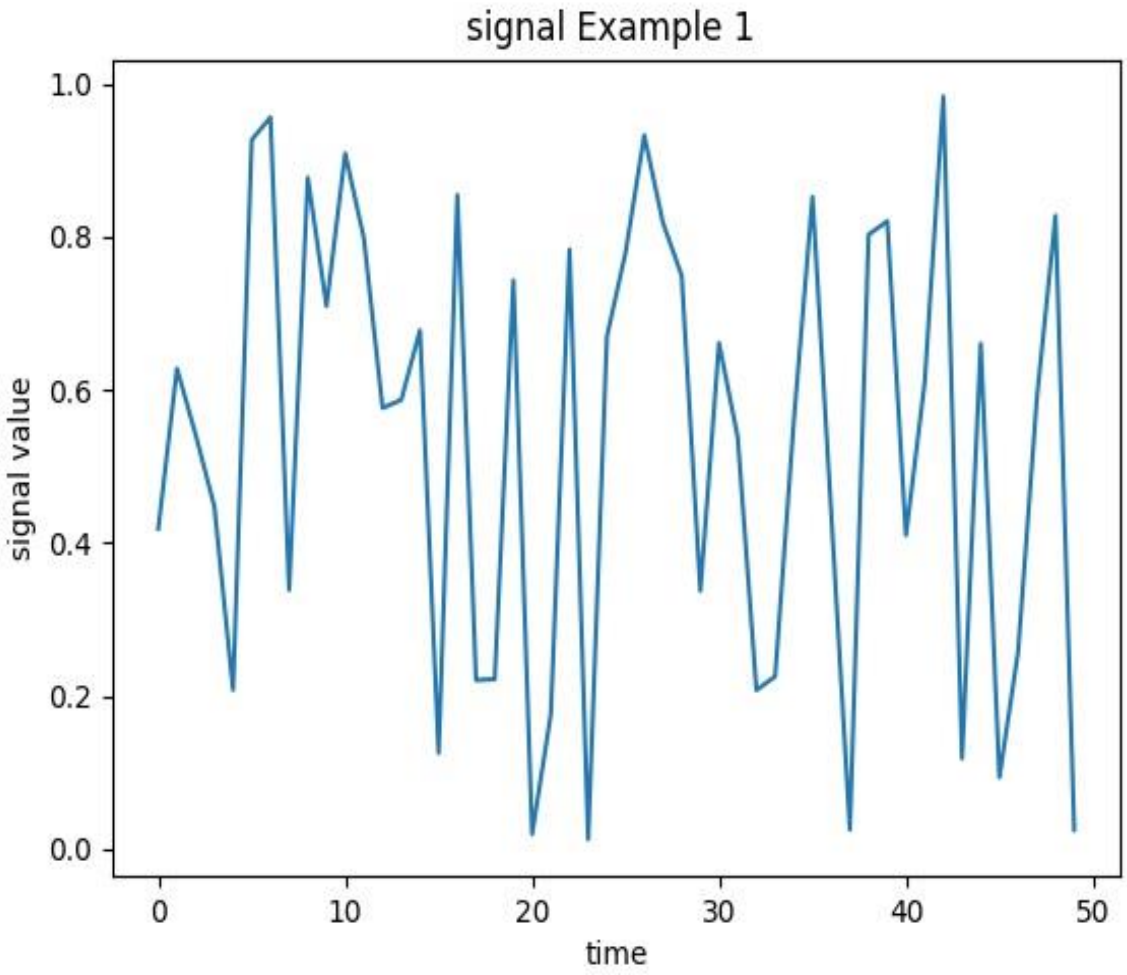
Section 3.1.1 – Synthetic Data:

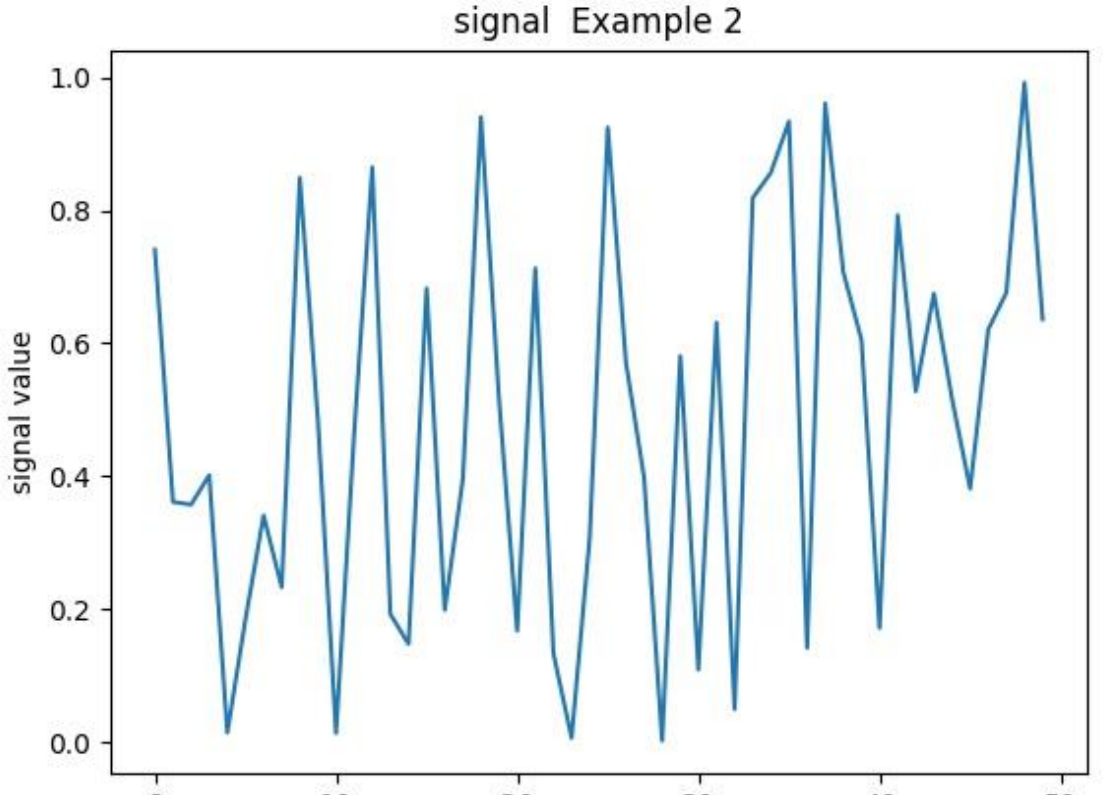
In this section we have generated random data, composed of 10,000 sequences, each containing T = 50 numbers.

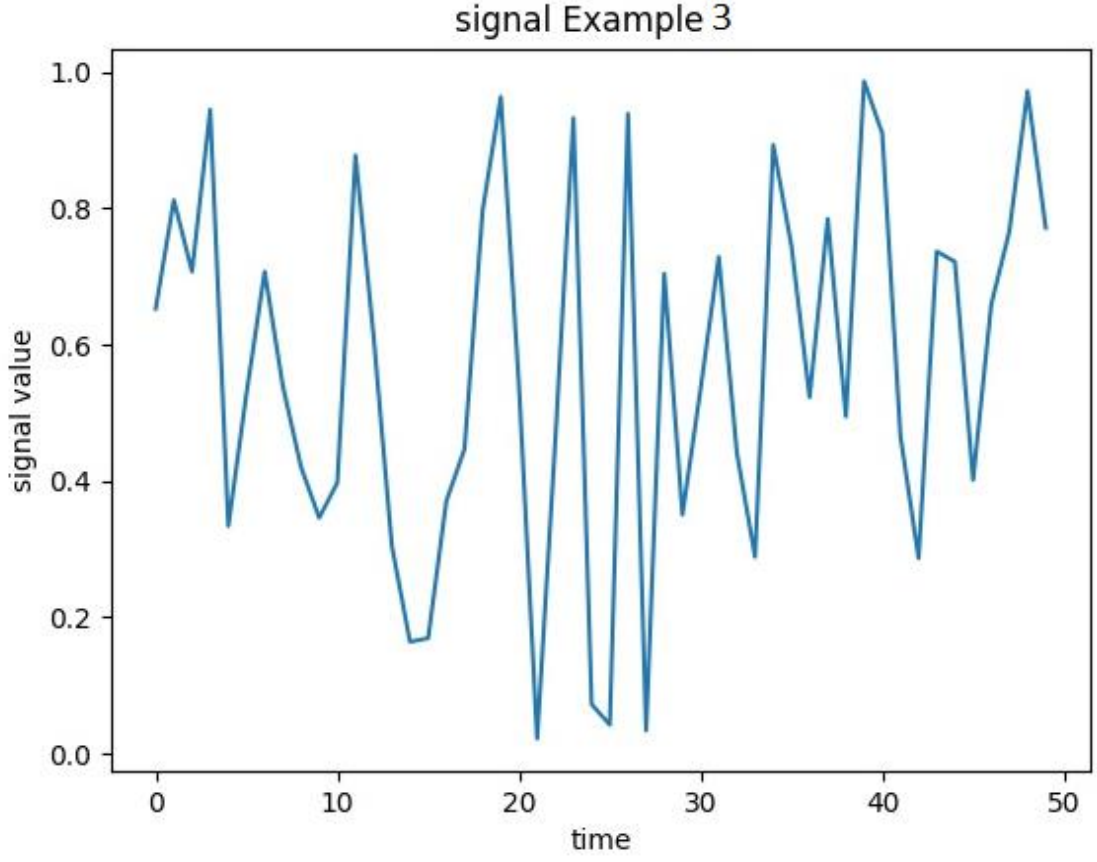
For each of the sequences marked $(x_1, \dots, x_{50})$, $x_i \in [0,1]$ and $\frac{1}{50}\sum x_i \simeq \frac{1}{2}$

The data was split into 3 data sets: train, validation and test, with ratios of 60%, 20% and 20% accordingly.

Presented here are 3 examples of the random signals we generated:




signal Example 2
signal value
1.0
0.8
0.6
0.4
0.2
0.0


signal Example 3
signal value
1.0
0.8
0.6
0.4
0.2
0.0
0
10
20
30
40
50
time

Section 3.1.2

For this section, we implemented an LSTM Auto Encoder:

LSTM_AE(input_size, hidden_size, output_size,Labels = 1)

For this task:

input_size = output_size = 1 (dimension of each examples in a sequence – in this case each $x_i \in R$).

The hidden_size is a parameter for the grid search performed.

The AE is composed of:

1.Encoder = LSTM model with input dimension of 1 (each feature x_i is a number)

The output of the encoder is the context vector marked z, which is defined is the last hidden state, sized hidden_size (parameter of the network).

2.Decoder = LSTM model with input dimension of hidden size, which receives as input the context z repeated T (sequence length) times.

Output is of same size as input.

3. Fully Connected Layer – takes the activation of dimension (batch,T,hidden_size) and transforms it to the dimension (batch,T,hidden_size, Labels). In our case Labels =1 and we receive an output of the same dimension as the original input for the net.

Grid search

When training, we performed a grid search for the optimal parameters, training the net with chosen parameters on the train data, and testing it on the validation data every 5 epochs. for each parameters set we ran 300 epochs The optimizer used was Adam.

The range of parameters examined was:

Learning rate = [0.001,0.01]

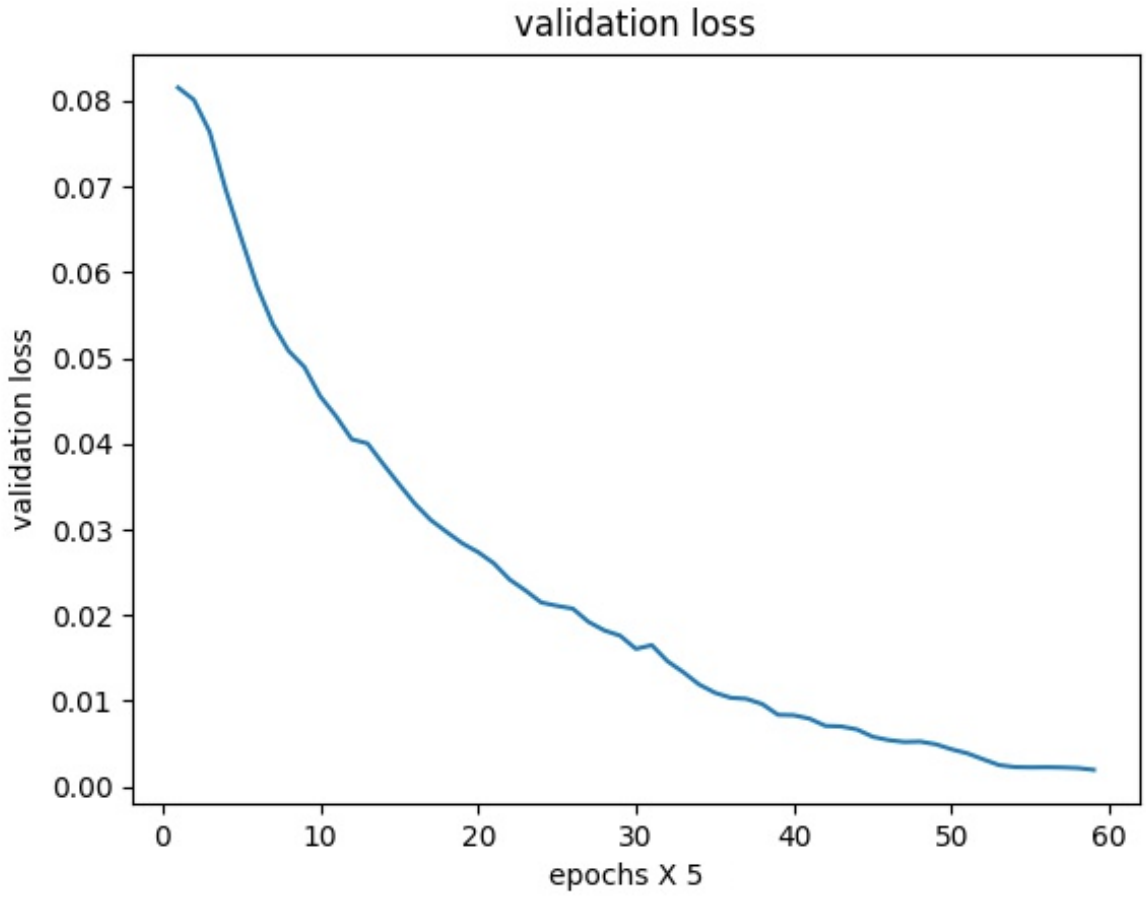
Gradient clipping = [1,0(None)]

Hidden size = [30,60,120].

Finally, the optimal parameters chosen were:

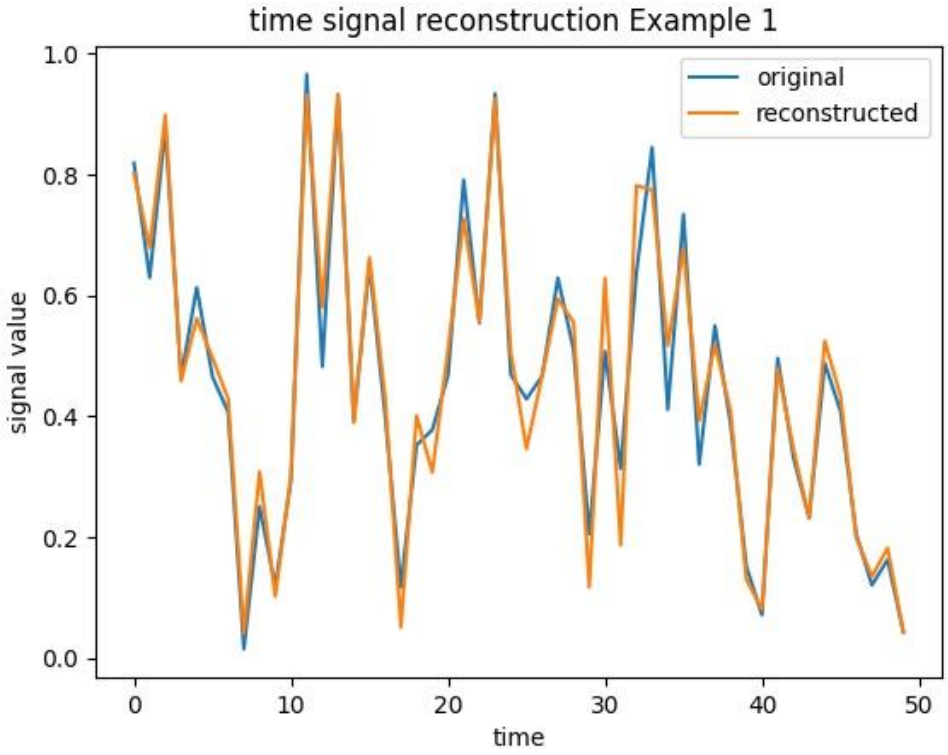
Learning rate = 0.001, gradient clipping = 1, Hidden size =60.

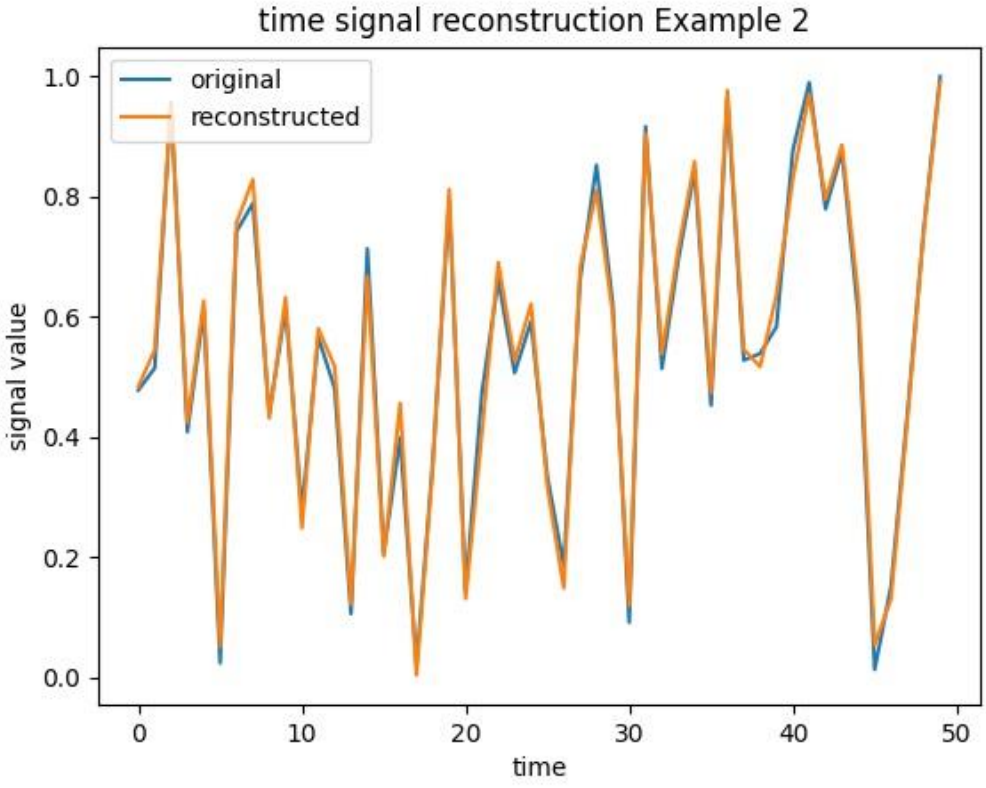
Below is the plot of the loss (MSE) calculated on the validation data each 5 epochs:



Finally, we ran the model trained with the chosen parameters on the test data, to receive the loss of 0.0004.

In the following figures, we present 2 examples of signals from the test data, reconstructed by our model:

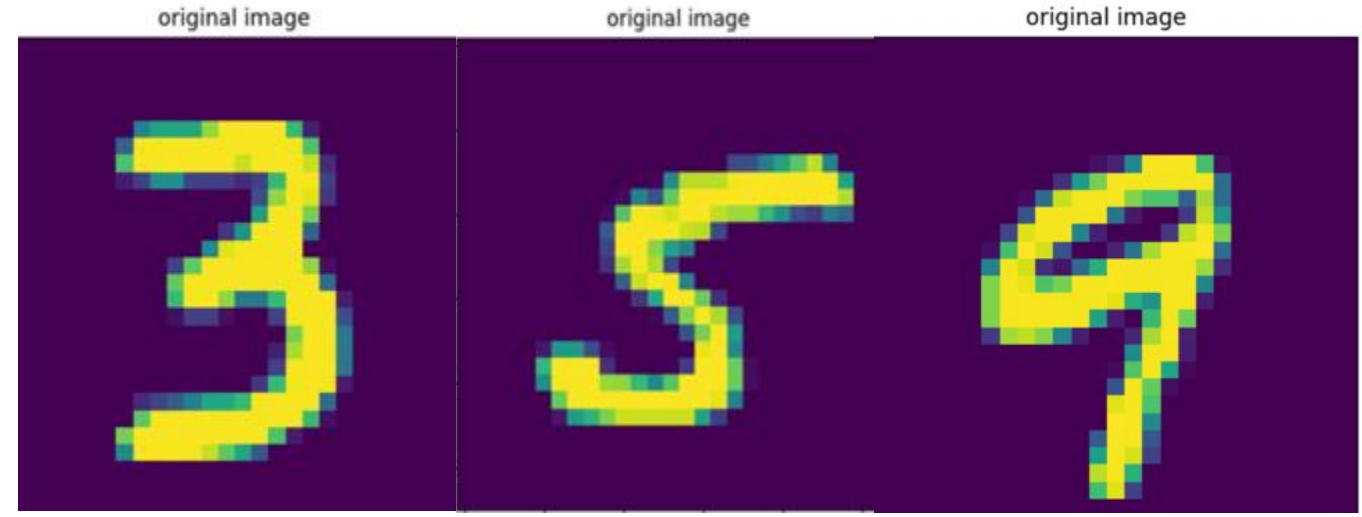


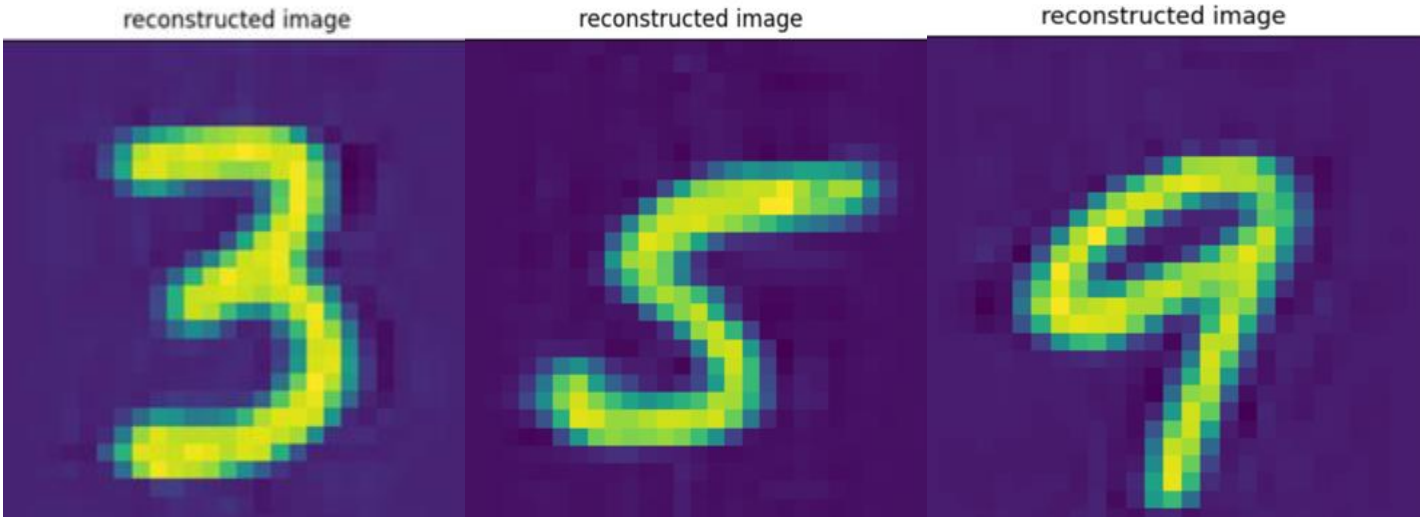


Section 3.2- MNIST

During this section, we trained both a reconstruction and classification AE models on MNIST data. Due to computational limitations (very high training time), we used each image as a sequence of 28 rows (feeding the AE row by row instead of pixel by pixel). We used the chosen parameters from the grid search of the last section.

3.2.1: presented in the next 3 figure pairs, are the reconstruction of digits our model performed on the test data, post training:





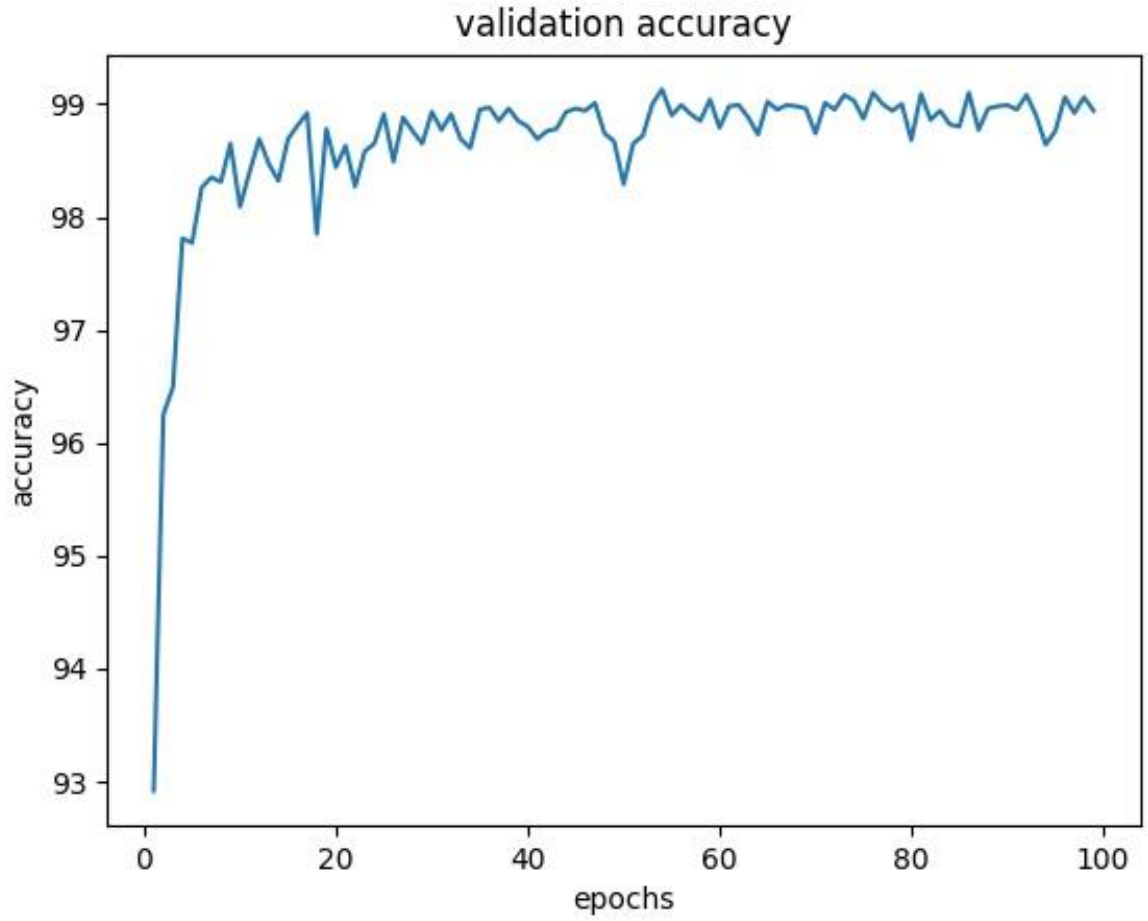
3.2.2 – Classification:

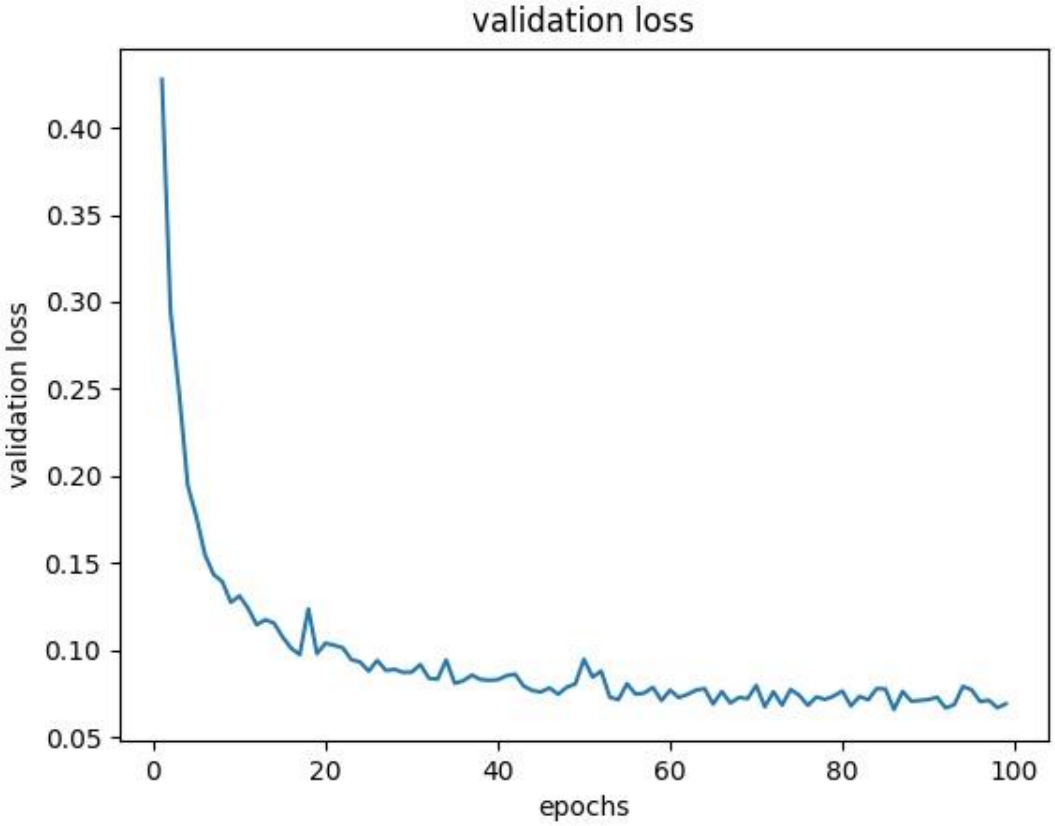
During this section we preformed classification on the images¸ meaning the last FC layer was changed to output a "probabilities" vector sized 10 for each input sequence¸ and we added to our MSE reconstruction loss a second Cross Entropy loss , making the loss for the entire network:

Total loss = $\frac{MSE(original\ X, reconstruced\ X) + CE(labels, predictions)}{2}$

Training was very fast, reaching over 98% of accuracy in less than 10 epochs.

Below we present graphs for both validation loss and accuracy vs epochs:




validation loss
validation loss
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0
20
40
60
80
100
epochs

Section 3.3 – S&P500

In this section, we performed reconstruction and classification on data of S&P500 stocks, working with the “high” (max) stock values per day.

Plotted below, we can see an example of the Amazon and Google high values of the stocks, between 01/14 – 01/18.

In order to normalize the data according to the requirements, we defined a Normalizer() class, performing the following normalization:

stock sequence rescaled = $\frac{stock\ sequence - \min(stock\ sequence)}{\max(stock\ sequence)}$, making the sequence values range in [0,1].

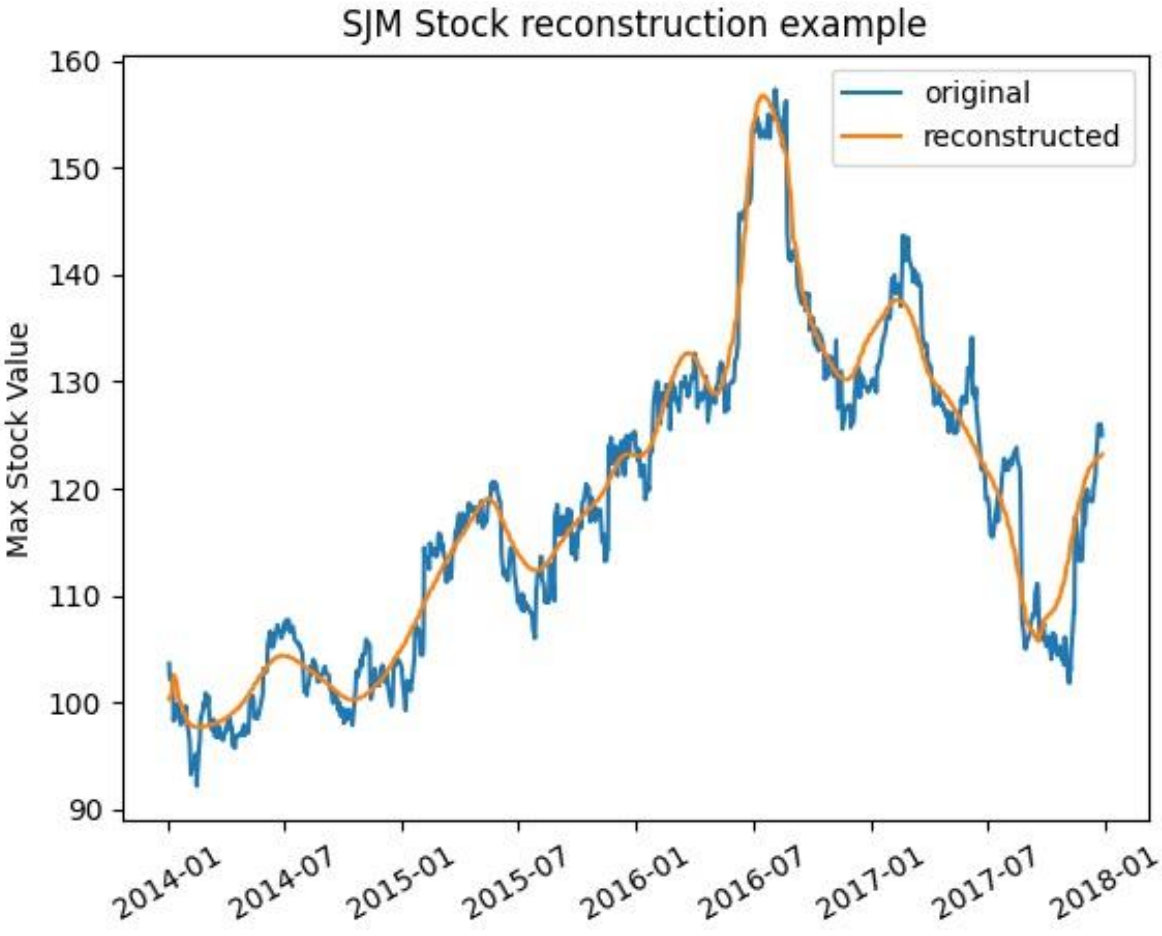
then normalized stock sequence = $\frac{\text{stock sequence rescaled}}{2 * mean(\text{tock sequence rescaled})}$

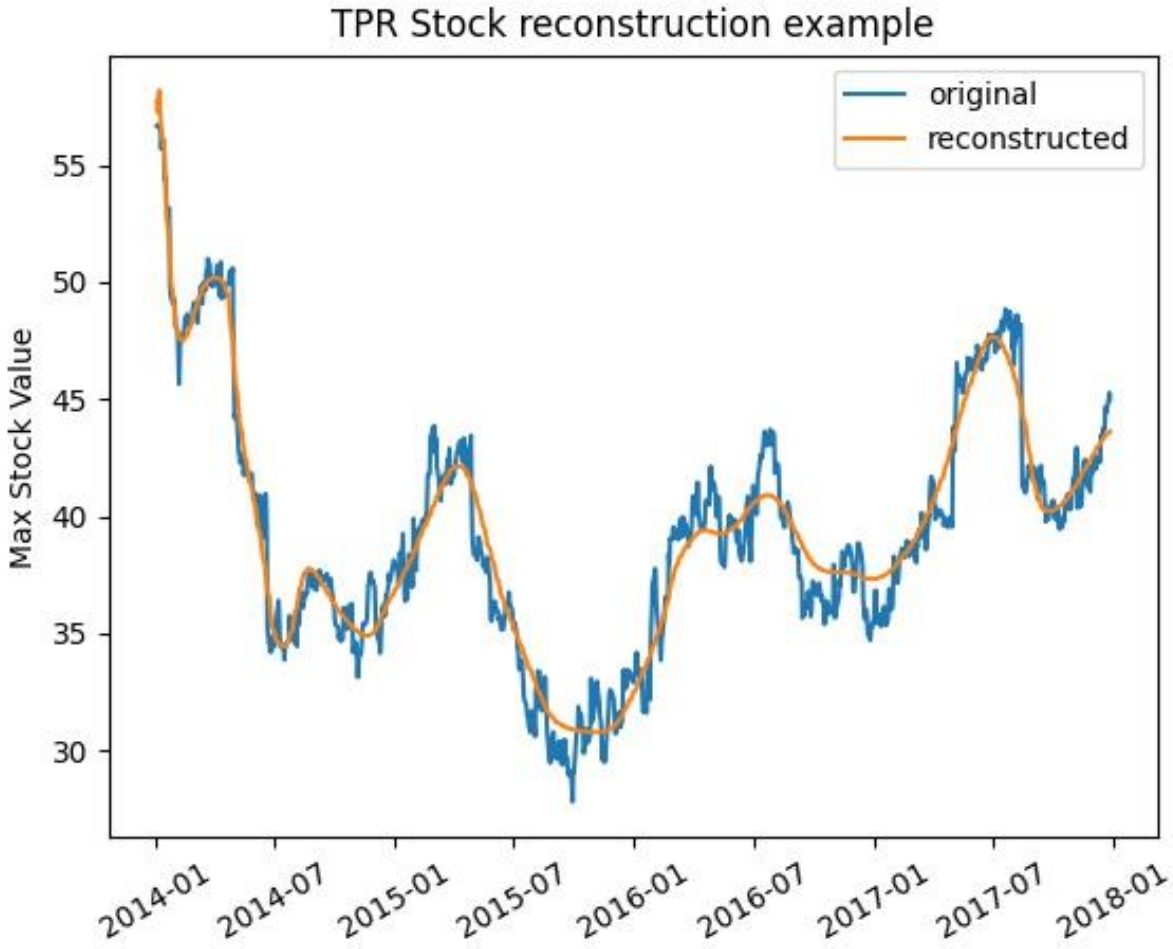
setting the mean as $\frac{1}{2}$. After reconstruction, we will apply the opposite transformation to unnormalize the reconstructed stocks back to the original scale.

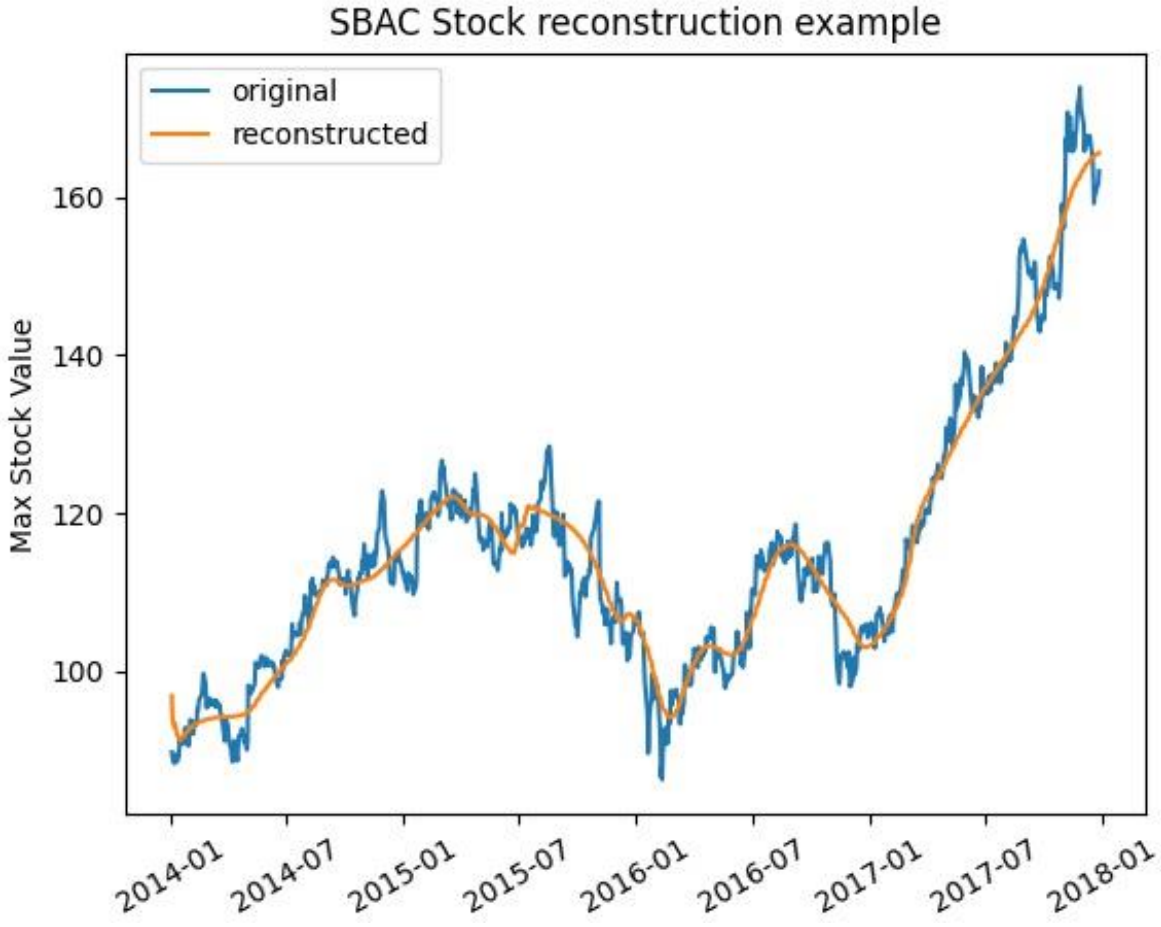
During training, we again used a validation data set to select optimal hyper parameters, finally choosing:

Hidden size = 120, learning rate = 0.001, gradient clipping =1.

The following 3 graphs present reconstruction of 3 different stocks from our test data, after training the model with chosen hyper parameters:




TPR Stock reconstruction example
original
reconstructed
Max Stock Value
55
50
45
40
35
30
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01


SBAC Stock reconstruction example
original
reconstructed
Max Stock Value
160
140
120
100
2014-01
2014-07
2015-01
2015-07
2016-01
2016-07
2017-01
2017-07
2018-01

3.3.3: in this section, in order to perform classification we added a FC layer for prediction, acting on the decoded output in parallel to the reconstruction FC layer. For each stock high value $x_1, \ldots . x_{T-1}$, we predicted $\hat{x}_2, \ldots, \hat{x}_T$, meaning the value of the next day. For the loss of the network, we defined:

$$\text{Loss} = \frac{MSE(stocks, reconstructed\ stocks) + MSE(predicted\ stocks, target\ stocks)}{2}$$

Below we present the plots of training and test loss vs epochs (3K epochs in total):