Homework Assignment

Student Information

Student Name	**Student ID**
Amit Aharoni	206790149
Raz Shmueli	203748645

Setup

Run the following from within the project working directory:

```
make all
source venv/bin/activate
```

This command will ensure that you have all the requirements and you build a local Python virtual environment including activating it.

Part 3.1 - Synthetic Data

3.1.1

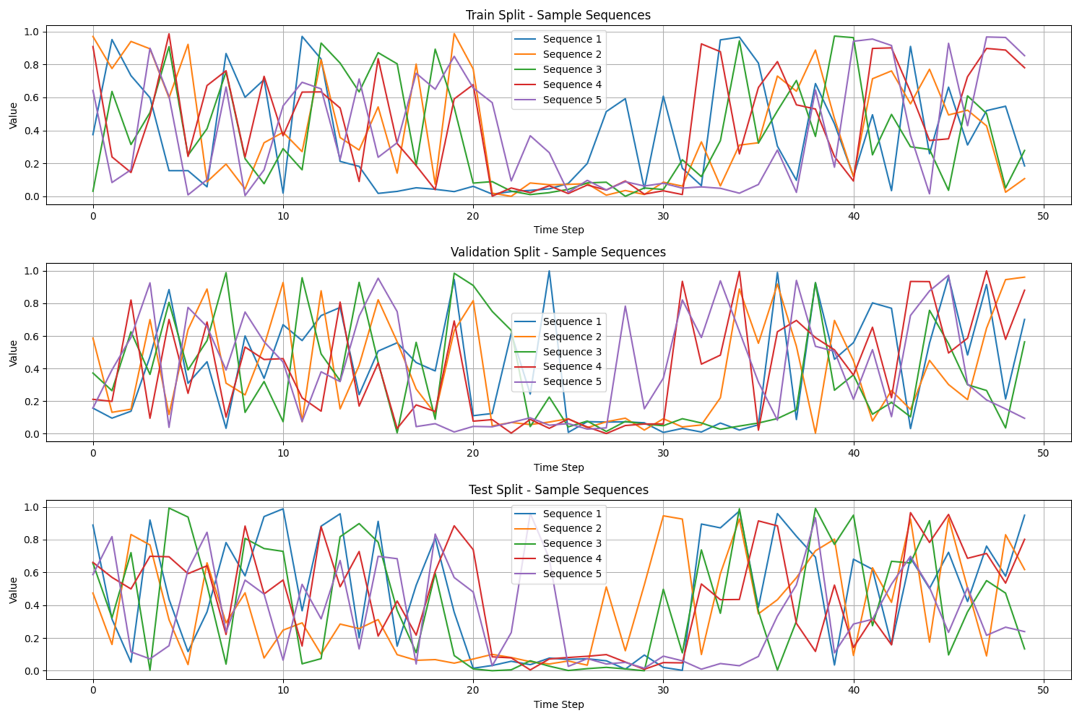


Figure 1: Example of synthetic data from each split - Train, Validation, Test

We can see that the different sets sampled from the same distribution. In addition, its importent to note, that the even though the different sequence are similar, there is clear variance that distinguish between all.

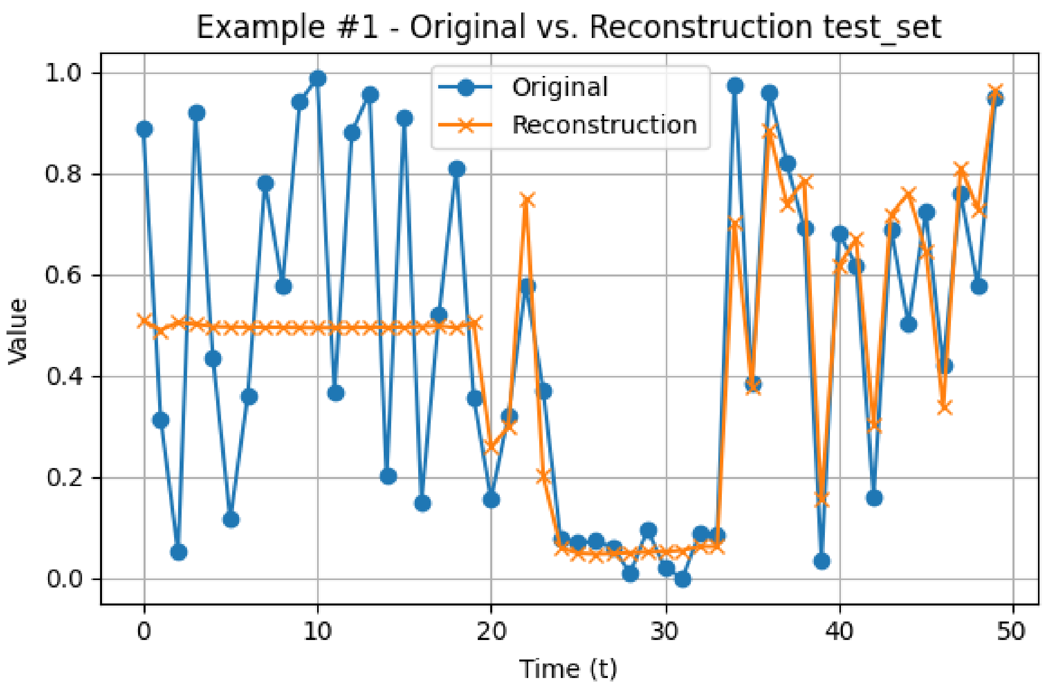
In order to create the above chart, rum

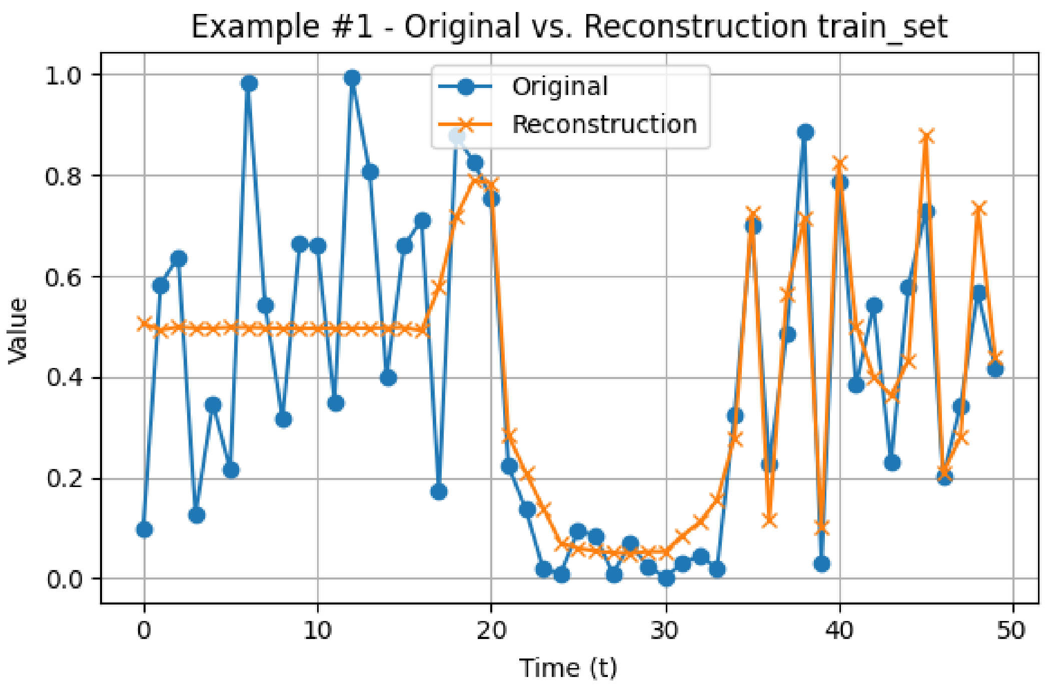
```
python RNNBasics/src/lstm_ae_toy_3_1/lstm_ae_toy_3_1_1.py
```

3.1.2.1

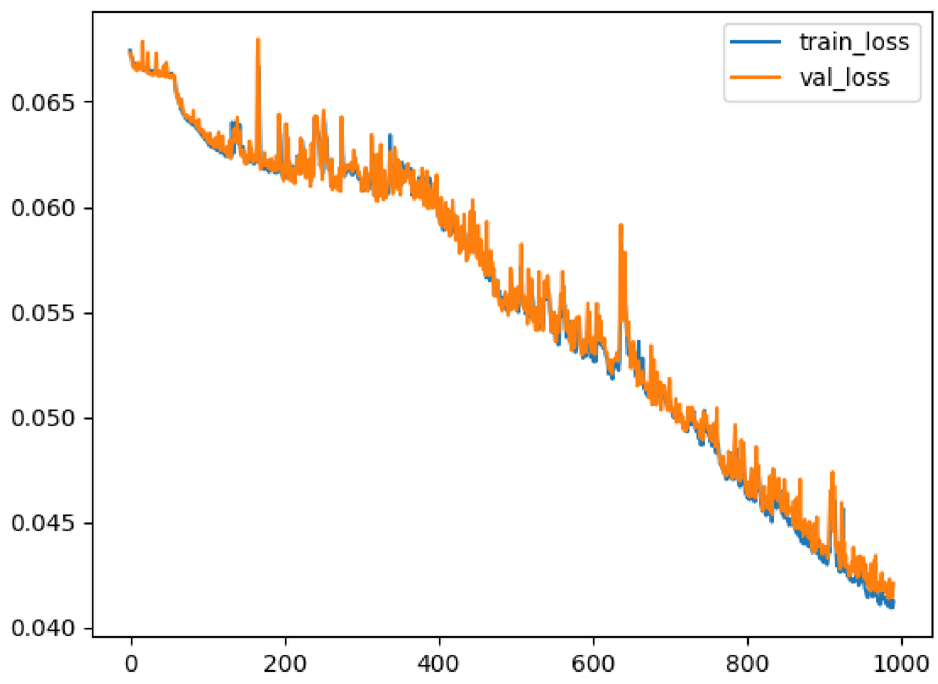
Plots for reconstruction on train and test:

3.1.1





Its evident that, for both the training and test sets, the model struggles to reconstruct the start of the sequence. However, this issue could potentially be resolved with additional iterations, as indicated by the loss curve below.



The loss trend suggests that we can further improve the performance by increasing the number of iterations. This may also help resolve the issue of poor reconstruction at the start of the sequence.

After conducting a grid search (see parameter intervals below), we found that over-parametrization led to over-fitting issues. Consequently, we selected a combination that resulted in a lower loss in the test set. We acknowledge the lower reconstruction performance on the training set, as shown in the plot.

In order to run GridSearch with OPTUNA package run:

```
python RNNBasics/src/lstm_ae_toy_3_1/lstm_ae_toy_3_1_2_train_without_grid_search.py
```

In order to run simple training on chosen parameters, run

```
RNNBasics RNNBasics/src/lstm_ae_toy_3_1/lstm_ae_toy_3_1_2_train_without_grid_search
```

An examples for intervals for grid serch we tested are :

```
DEFAULT_HIDDEN_SIZE_MIN = 32
DEFAULT_HIDDEN_SIZE_MAX = 49
DEFAULT_LR_MIN = 1e-4
DEFAULT_LR_MAX = 1e-2
DEFAULT_GRAD_CLIP_MIN = 0.1
DEFAULT_GRAD_CLIP_MAX = 5.0
DEFAULT_NUM_LAYERS_MIN = 1
DEFAULT_NUM_LAYERS_MAX = 20
```

In order to see all default configuration for optuna and final model see

```
RNNBasics/src/lstm_ae_toy_3_1/config.py
```

Part 3.2

3.2.1

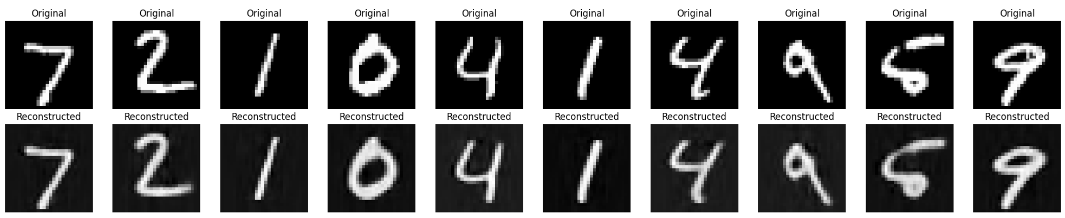


Figure 2: First row : Real digits. Second row : Reconstruction, row by row

It looks like the model learned the row-conditioned distribution of the MNIST digits. However, it is clear that there is a small gap that appears due to the blurry reconstruction.

In order to train model and reconstruct with grid-search run

```
python RNNBasics/src/mnsit_reconstruction_3_2/reconstruct_mnist_by_row_3_2_1.py
```

3.2.2

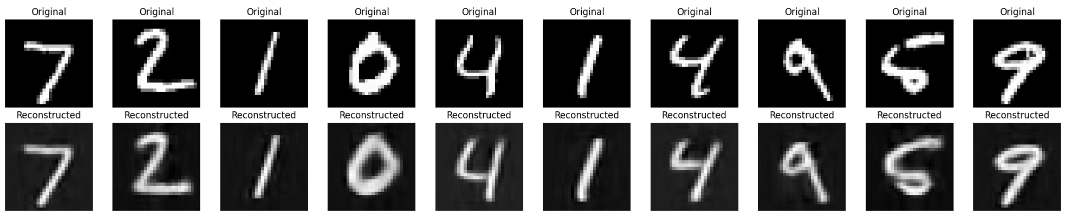


Figure 3: First row : Real digits. Second row : Reconstruction, row by row with classification task

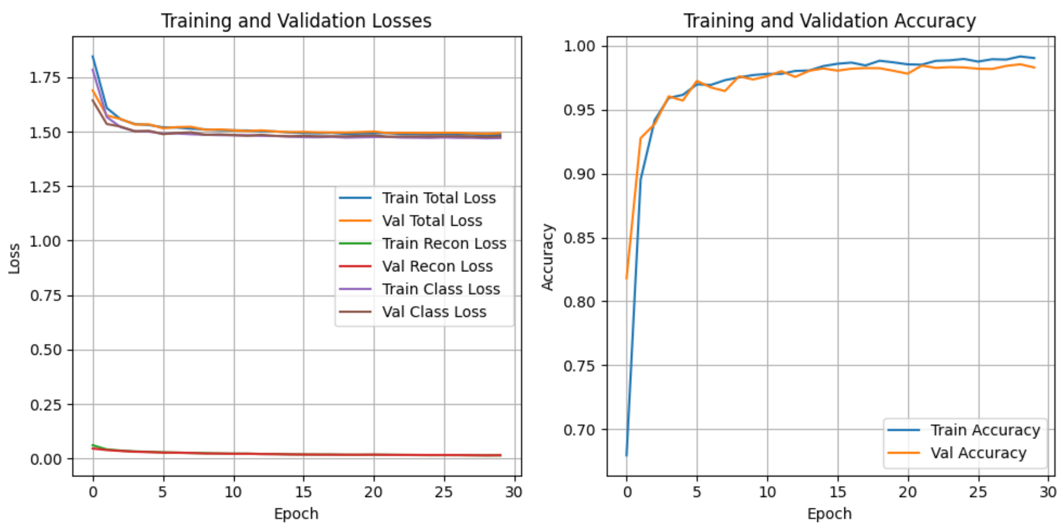


Figure 4: Left : reconstruction and classification loss for all dataset parts Right:Accuracy of train vs val

As same as above, here the LSTM architecture achieves impressive performance. In addition, we noticed that the convergence of the reconstruction loss for the combined task of reconstruction was faster. An intuition for this could be that the two losses work together to predict the correct next step.

3.2.3

For pixel-by-pixel reconstruction, the results show lower performance and that occur due to the lack of resources. This is evident by observing the accuracy vs. epoch plot below, which displays a sharp accuracy improvement curve, which could push performance up by by simply continuing training.

Two interesting points to note:

During the first 5–7 epochs, the model shows no improvement at all, and the sharp increase starts from epoch 8. Although the accuracy rises sharply, we do not see the same trend in the loss curves. This may indicate that for the task of classification, it is necessary to be closer to the extreme in order to make correct predictions.

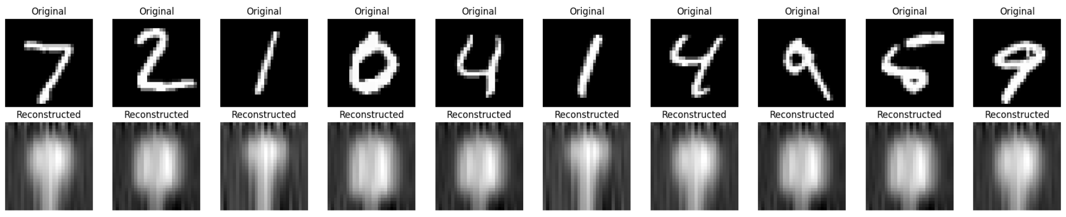


Figure 5: Pix by Pix reconstruction

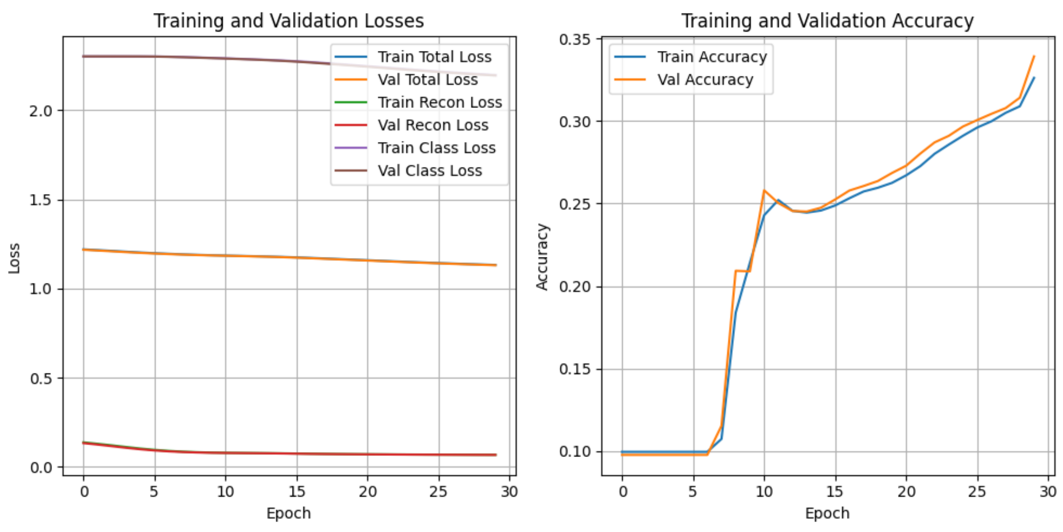


Figure 6: reconstruction Pix by Pix and classification loss for all dataset parts Right:Accuracy of train vs val

One technical aspect that greatly helped us to achieve convergence (as we initially struggled with it) was assigning a higher weight to the reconstruction loss(0.7). See the code below.

Part 3.3

3.3.1

As its clearly shown, the AMZN (Amazon) and GOOGL (Google) stocks are increasing over a period of 4 years.

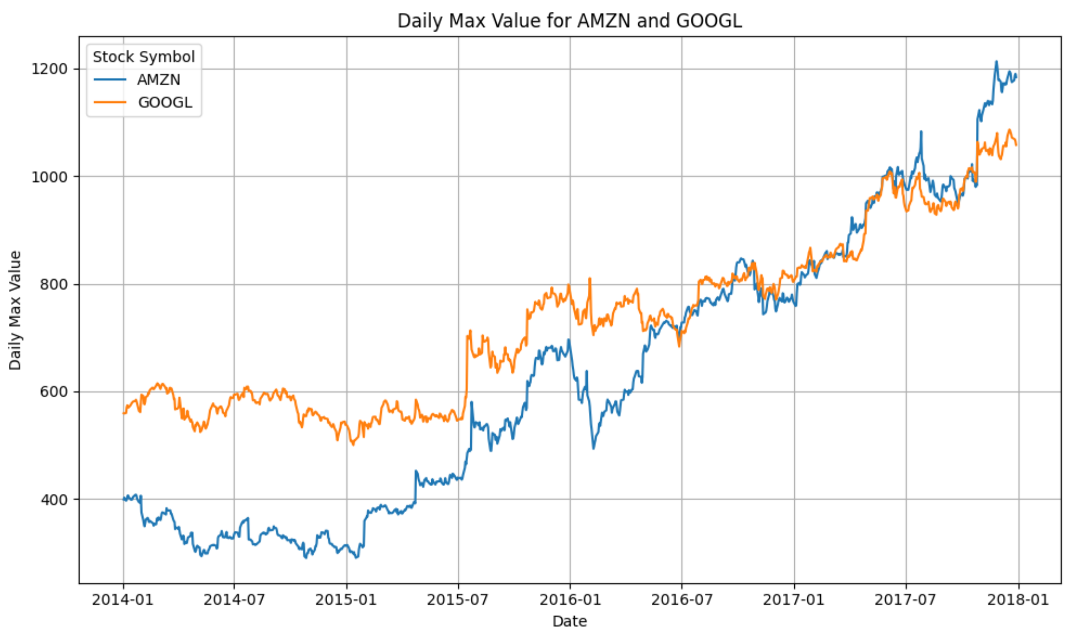


Figure 7: Daily max value plot over time for amazon and google stocks

To get these plots, run:

```
python RNNBasics/src/snp_reconstruction_3_3/reconstruct_snp_graph_3_3_1.py
```

3.3.2

Overall, the three stocks show relatively good reconstructions with the LSTM Autoencoder (AE), although they are not reproduced perfectly. In particular, the model successfully captures the general upward or downward trends of each stock. This indicates that the latent representation learned by the LSTM AE is able to encode the broad temporal dependencies in the price data.

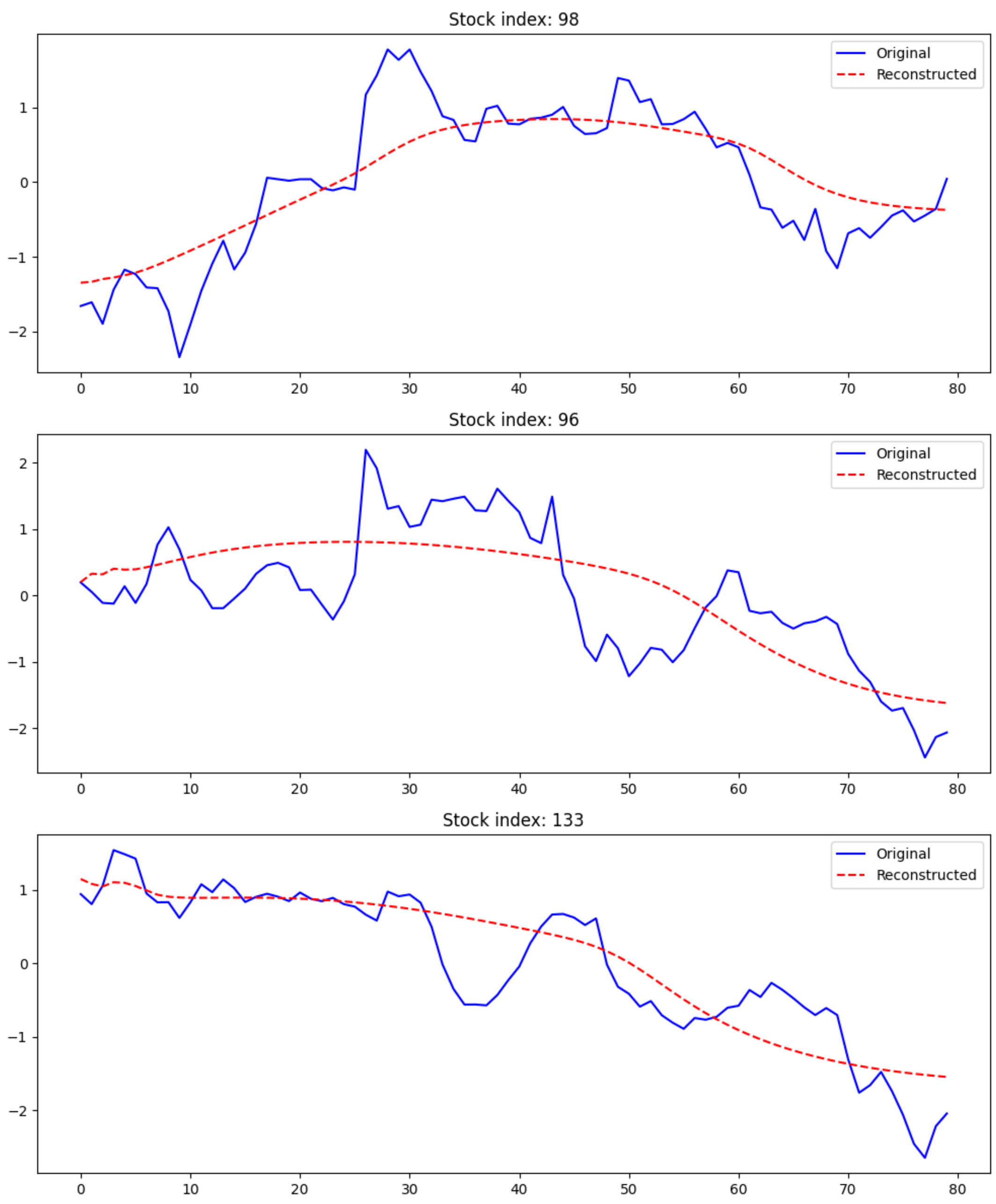


Figure 8: Example of 3 different reconstructed stocks

```
python RNNBasics/src/snp_reconstruction_3_3/reconstruct_snp_train_3_3_2.py
```

3.3.3

The model shows learning abilities in both prediction and reconstruction loss. As mentioned in the previous section, with more iterations, the model will reach better results based on the trend of the loss plot.

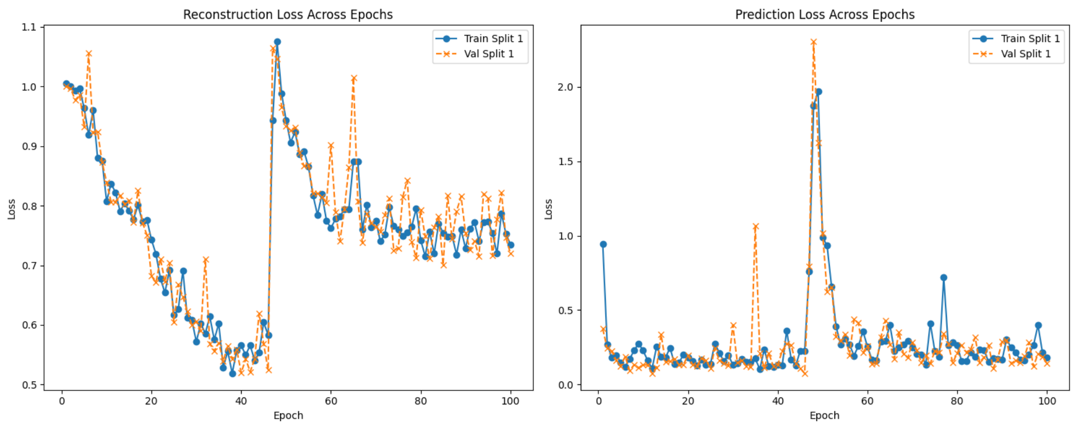


Figure 9: Reconstruction and Prediction loss vs Epochs

```
python RNNBasics/src/snp_reconstruction_3_3/reconstruct_and_predict_snp_3_3_3.py
```

3.3.4

Obviously, the results from our LSTM Autoencoder are not perfect—if they were, we would all be rich. Still, the model demonstrates a meaningful ability to learn and reconstruct the underlying trend in each of the stock price series.

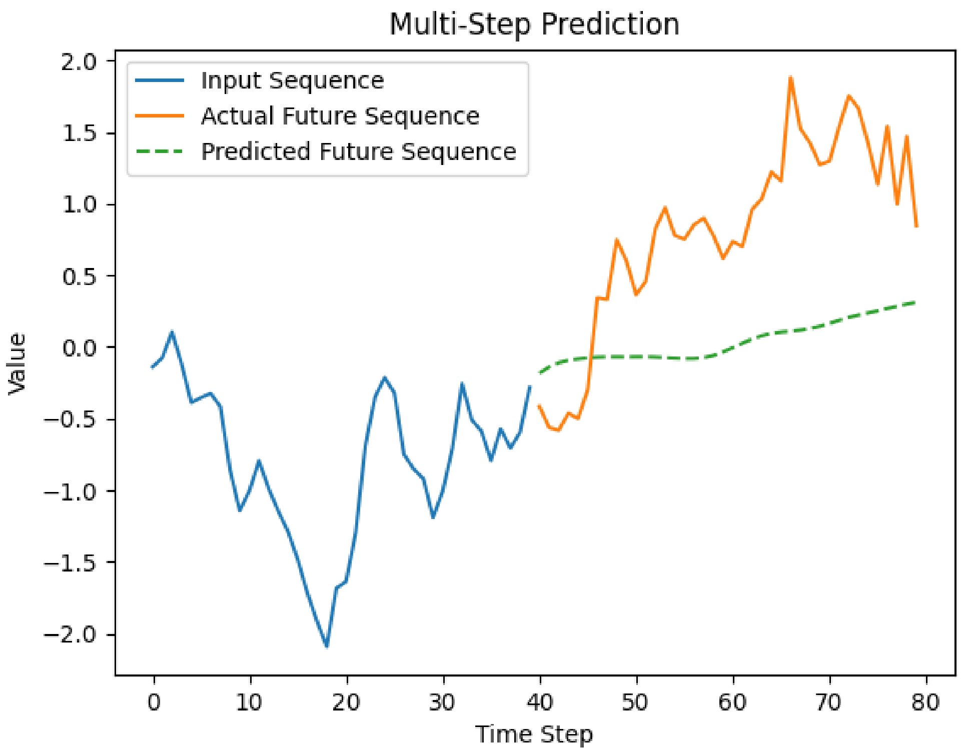


Figure 10: Prediction of next steps.

```
python RNNBasics/src/snp_reconstruction_3_3/predict_next_steps_snp_3_3_4.py
```