CAP 6615 - Neural Networks
Programming Assignment-4
Recurrent Neural Network

Venkata Vynatheya Jangal, Sohith Raja Buggaveeti, Upendar Penmetcha,
Chinmai Sai Eshwar Reddy Kasi and Venkateswarlu Tanneru

University of Florida

Friday, 8th April 2022

Contents

Recurrent Neural Network

1 Abstract:

The Recurrent Neural Network (RNN) model is used in this assignment. A Recurrent Neural Network (RNN) is a type of artificial neural network in which connections between nodes create a directed graph along a temporal sequence. It's a type of generalized feed forward network with built-in memory for processing input data. A convolution neural network (CNN) is also employed in addition to the RNN. A convolutional neural network is made up of three layers: an input layer, hidden layers, and an output layer. Because the activation function and final convolution mask the inputs and outputs of middle layers in any feed-forward neural network, they are referred to as hidden layers. The hidden layers of a convolutional neural network contain layers that perform convolutions. This is followed by other layers such as pooling layers,fully connected layers, and normalization layers.In this assignment,a recurrent neural network(RNN) is designed and trained on S&P 500 data .The model is tested by predicting the next one,two,three or four values using a sliding sampling window of 180 days.The models are also re-tested on noise-corrupted data to observe the performance. The goal of this project is to optimize the RNN to give the best possible results.

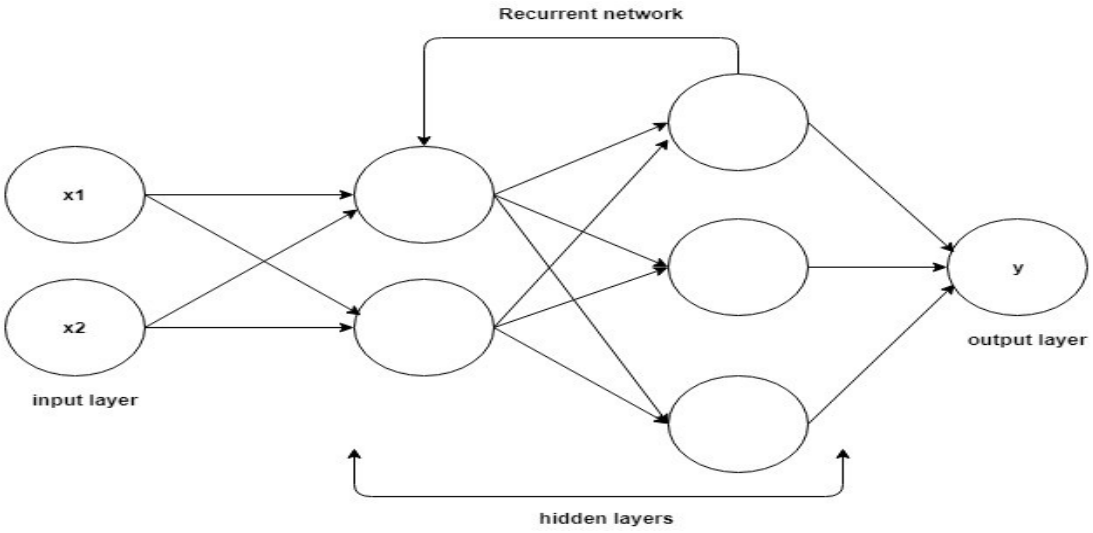

Figure 1: Recurrent Neural Network

2 Dataset Generation:

An S&P 500 dataset between January 4, 1960 and December 31, 2021 was gathered from the Yahoo Finance website to be used as input for this assignment. The table is scraped from the website and was saved as a csv file. One of the standard indicators used to determine whether a market is overpriced, undervalued, or correctly valued is the Shiller P/E ratio. The Shiller-PE website has it available for download. From 1960 to 2021, the Shiller P/E ratio data was gathered monthy. To transform this monthly data to daily data, we apply the linear interpolation approach. The data from the S&P 500 and the Shiller P/E ratio were then combined and saved in S&P500.csv. After then, the data was split into training and test sets.

	Date	Open	High	Low	Close	Adj Close	Volume	S&P 500 PE Ratio
0	1960-01-04	59.91	59.91	59.91	59.91	59.91	3990000	17.055161
1	1960-01-05	60.39	60.39	60.39	60.39	60.39	3710000	17.033548
2	1960-01-06	60.13	60.13	60.13	60.13	60.13	3730000	17.011935
3	1960-01-07	59.69	59.69	59.69	59.69	59.69	3310000	16.990323
4	1960-01-08	59.50	59.50	59.50	59.50	59.50	3290000	16.968710
...	...	...	...	...	...	...	...	...
15602	2021-12-27	4733.99	4791.49	4733.99	4791.19	4791.19	2264120000	28.820000
15603	2021-12-28	4795.49	4807.02	4780.04	4786.35	4786.35	2217050000	29.120000
15604	2021-12-29	4788.64	4804.06	4778.08	4793.06	4793.06	2369370000	29.420000
15605	2021-12-30	4794.23	4808.93	4775.33	4778.73	4778.73	2390990000	29.720000
15606	2021-12-31	4775.21	4786.83	4765.75	4766.18	4766.18	2446190000	30.020000

15607 rows × 8 columns

Figure 2: Dataset

The resulting dataset is an N-dimensional time series data indicated by the symbols P_0,P_1,...,P_{N-1}. The whole dataset is split into 180 days (6 months) fixed-size windows. As a result, whenever the window is pushed to the right by size W, the data in it is pushed to the right as well. The sliding windows do not overlap in any way. After that, the data is normalized to the range [0,1]. After that, the data is separated into test and training datasets.

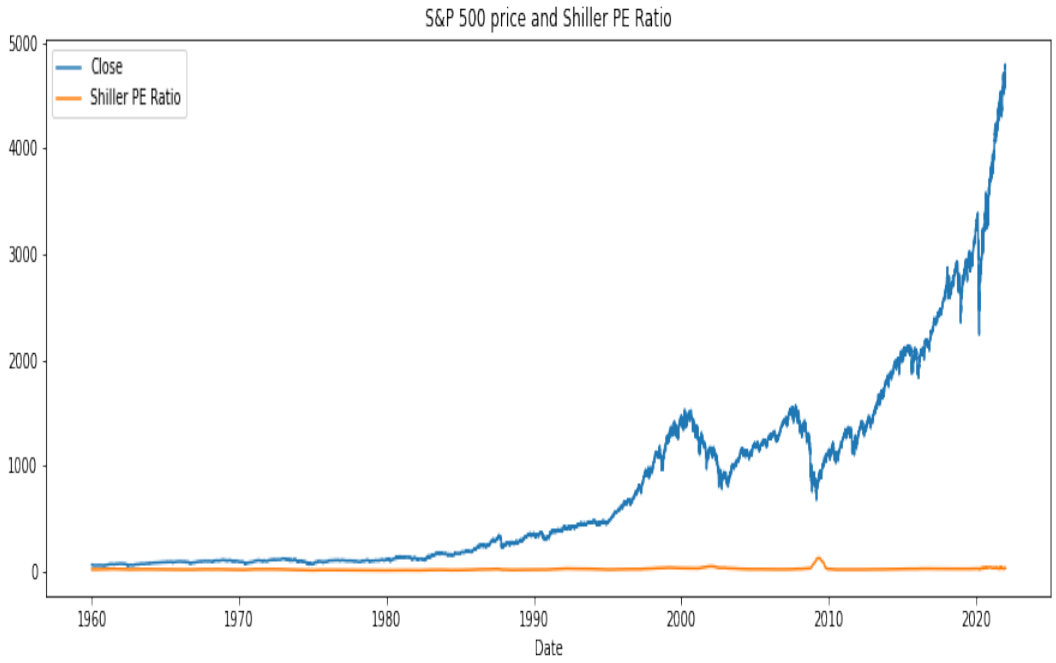

Figure 3: Graph of Dataset with Shiller P/E ratio

3 Network Parameters

3.1 RNN

1. **Input:** Generated Dataset S&P 500 data from Jan 1^{st},1960 to Dec 31^{st},2021.
2. **Predicted Output feature:** Close price
3. **Activation function:** ReLU
4. **Epochs:** 100
5. **Optimizer:** Adam
6. **Evaluation metrics:** prediction error, accuracy
7. **Number of time steps:** 180
8. **Loss metrics:** Mean squared error

3.2 CNN

1. **Input:** A-B-C-D Curve Images
2. **Predicted Output feature:** Probabilities Derived from A-B-C-D Curves
3. **Activation function:** ReLU
4. **Epochs:** 50
5. **Optimizer:** Adam
6. **Evaluation metrics:** prediction error, accuracy
7. **Loss metrics:** Cross - Entropy

The model is initially trained on the CNN with A-B-C-D curve pattern images as input, and then the CNN output is fed into the RNN, which produces price predictions. The model is trained on all of the data from 1960 to 2014 and then tested on data from 2015 to 2020. The RNN that was created is a regression model that predicts future Stock Close Prices. The anticipated price output is then compared to the original price data in order to evaluate the model. In comparison to the unoptimized RNN+CNN model, the optimized RNN+CNN model has a greater accuracy. The Adam optimizer was used as the activation function, along with ReLU. The optimized CNN+RNN performed significantly better once these changes were made.

4 RNN Model

The LSTM model is the architecture used to deploy this model. Long Short Term Memory (LSTM) is an acronym for Long Short Term Memory. Long Short-Term Memory (LSTM) networks are a modified version of recurrent neural networks that make it simpler to recall information from the past. Here, the RNN's vanishing gradient problem is solved. Given time lags of uncertain duration, LSTM is well-suited to identify, analyse, and predict time series. Back-propagation is used to train the model. Three gates are present in an LSTM network:

(i) **Input gate:** discover which value from input should be used to modify the memory. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from -1 to 1

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$
$$\tilde{C}_t = \tanh\left(W_C \cdot [h_{t-1}, x_t] + b_C\right)$$

(ii) **Forget gate:** Identify which details in the block should be removed. The sigmoid function determines this. It examines the previous state (h t-1) and the content input(Xt) and outputs a number between 0 and 1 for each number in the cell state C t1.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right)$$

(iii) **Output Gate:** the output of the block is determined by the input and memory of the block. The Sigmoid function determines which values are allowed to pass through 0,1, and the tanh function assigns weightage to the values that are passed, ranging from -1 to 1, and is multiplied by the Sigmoid output.

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

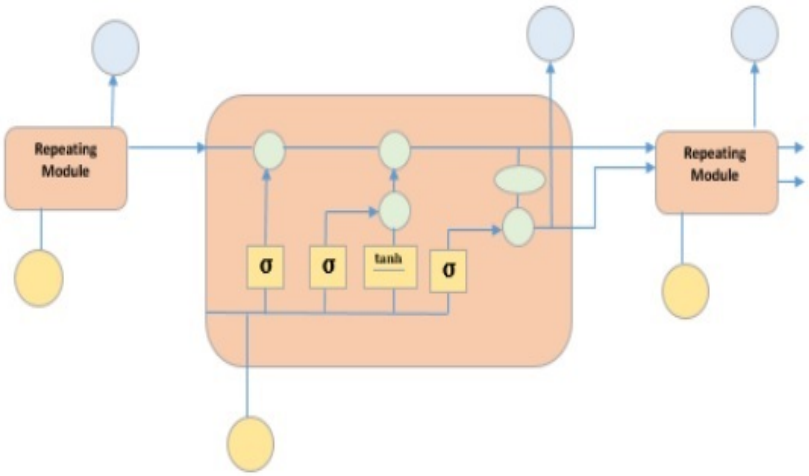

Figure 4: Sample LSTM Architecture

4.1 Loss Metrics:

The model uses mean squared error as a loss metric. Mean squared error is used to tell how close a regression line is to a set of points. It also gives more weight to larger differences.
One parameter for evaluating classification models is accuracy. Informally, accuracy refers to the percentage of correct predictions made by our model. The following is the formal definition of accuracy:

Accuracy = (Number of correct predictions) / Total Number of predictions

Mean squared Error(MSD): In regression circumstances where your expected and predicted outcomes are real-number values, mean squared error is used.The loss is calculated using a simple formula. It's simply the difference in squares between the expected and predicted values.

$$MSD = \sum_{i=1}^{D} (x_i - y_i)^2$$

4.2 Activation Function:

1. **ReLU:** The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero.

$$f(x) = x^+ = \max$$

4.3 Optimizer:

Adam can be thought of as a hybrid of RMSprop and Stochastic Gradient Descent, with the addition of momentum. It, like RMSprop, uses squared gradients to scale the learning rate and takes advantage of momentum by using the gradient's moving average.

- **Adam Optimizer:**The Adam optimizer employs a hybrid of two gradient descent techniques: Momentum: By taking into account the 'exponentially weighted average' of the gradients, this approach is utilized to speed up the gradient descent algorithm.
- **Adamax Optimizer:** The Adamax algorithm is implemented by this optimizer. It's an Adam version that uses the infinite norm. The default parameters are the same as those in the paper. In some cases, Adamax outperforms Adam, particularly in models containing embeddings.
- **SGD Optimizer:**The iterative approach of stochastic gradient descent is used to optimize an objective function with sufficient smoothness criteria. Because it replaces the real gradient with an estimate, it can be considered a stochastic approximation of gradient descent optimization. This minimizes the computing cost, especially in high-dimensional optimization problems, allowing for faster iterations in exchange for a reduced convergence rate.

Reason for Choosing Adam Optimizer:

- One interesting and dominant argument about optimizers is that SGD better generalizes than Adam but Adam has a greater speed.
- Picking the right optimizer with the right parameters, can help you squeeze the last bit of accuracy out of your neural network model.
- Adam uses Momentum and Adaptive Learning Rates to converge faster.

5 CNN Model:

Convolutional neural networks are image recognition systems that recognize complicated patterns and attributes. It captures basic features and patterns by dividing an image into many overlapping perceptual fields and running a variety of trainable filters through them. This process is performed numerous times, and as the image is filtered through more filters, deeper and more relevant features are retrieved and quantified. For example, to recognize an image of a car, we might have numerous filters that are sensitive to wheels, windows, exhaust pipes, or license plates, and the outputs of all of these filters are collated and quantified into a final classifier. CNN extracts usable and deep information by stacking numerous filters on top of each other to construct complicated feature-sensitive filters; if stock data was regarded as images, CNN can be used to extract useful and deep information.

5.1 CNN Model Summary:

```
(cnn_layers): Sequential(
  (0): Conv2d(1, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (1): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (2): ReLU(inplace=True)
  (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (4): Conv2d(4, 4, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (5): BatchNorm2d(4, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (6): ReLU(inplace=True)
  (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
)
(linear_layers): Sequential(
  (0): Linear(in_features=8100, out_features=2, bias=True)
)
```

Figure 5: CNN Model

6 RNN Model:

RNNs are a type of neural network that is both powerful and robust, and they are one of the most promising algorithms in use due to the fact that they are the only ones with an internal memory.

RNNs may use their internal memory to recollect important data about the input they receive, allowing them to predict what will happen next with excellent accuracy. This is why they're the algorithm of choice for time series, speech, text, financial data, audio, video, weather, and a variety of other sequential data types. Recurrent neural networks, in compared to other algorithms, can learn a lot more about a sequence and its context.

6.1 RNN Model Summary:

```
RNN(
  (rnn): RNN(3, 16, batch_first=True)
  (fc): Linear(in_features=16, out_features=4, bias=True)
)
```

Figure 6: RNN Model

7 Python Code of RNN + CNN :

7.1 Code for Data Generation:

The following code sample shows how to use the Pandas library to convert data from the Yahoo Finance website into a dataframe.

```
self.cnn_layers = nn.Sequential(
    nn.Conv2d(1, 4, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(4),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=2, stride=2),
    nn.Conv2d(4, 4, kernel_size=3, stride=1, padding=1),
    nn.BatchNorm2d(4),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(kernel_size=2, stride=2),
)
```

Figure 7: CNN Python Code

RNN Model

```
class RNN(nn.Module):
    def __init__(self, input_size, output_size, hidden_dim, n_layers):
        super(RNN, self).__init__()
        self.hidden_dim = hidden_dim
        self.n_layers = n_layers
        self.rnn = nn.RNN(input_size, hidden_dim, n_layers, batch_first=True)
        self.fc = nn.Linear(hidden_dim, output_size)

    def forward(self, x):

        batch_size = x.size(0)
        hidden = self.init_hidden(batch_size)
        out, hidden = self.rnn(x, hidden)
        out = self.fc(out)
        return out[:, -1, :], hidden

    def init_hidden(self, batch_size):
        hidden = torch.zeros(self.n_layers, batch_size, self.hidden_dim)
        return hidden

all_img = []
for i in range(X_full.shape[0]):
    x = X_full[i,:, 0]
    plt.axis('off')
    plt.plot(x, color ="black")
    figure = plt.gcf()
    figure.set_size_inches(8, 8)
    figure.canvas.draw()

    width, height = figure.get_size_inches() * figure.get_dpi()
    mplimage = np.frombuffer(figure.canvas.tostring_rgb(), dtype='uint8').reshape(576, 576, 3)
    gray_image = color.rgb2gray(mplimage)
    img_resize = resize(gray_image, (180, 180),anti_aliasing=True)
    img_resize = img_resize.astype('float32')
    plt.clf()
    all_img.append(img_resize)

all_x = np.array(all_img)
```

Figure 8: Python Code

7.2 Model Initialization and training

The model has been initialized with a Convolution 1D layer, followed by max pooling, and convolution 1D layers, as seen in the code sample below. ReLu are the activation functions that have been employed. Batch normalization has also been implemented. The Adam optimizer has also been utilized. DropOut has been implemented as well.

7.3 Correlation of stock data and Schiller P/E Ratio :

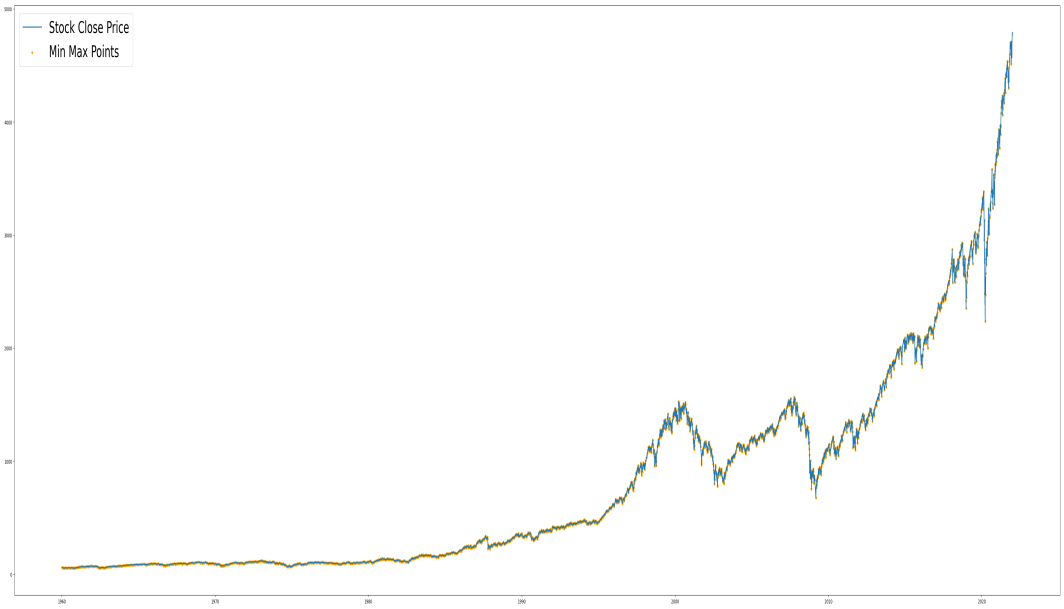

Figure 9: Correlation Graph

7.4 Generating Predictions:

On the basis of the test results, projections have been formed. After that, the score and accuracy are determined. Precision and recall are also taken into account.

8 Studying trends in stock close price with respect to Schiller P/E Ratio :

8.1 Predictions Graph on Unoptimized Model

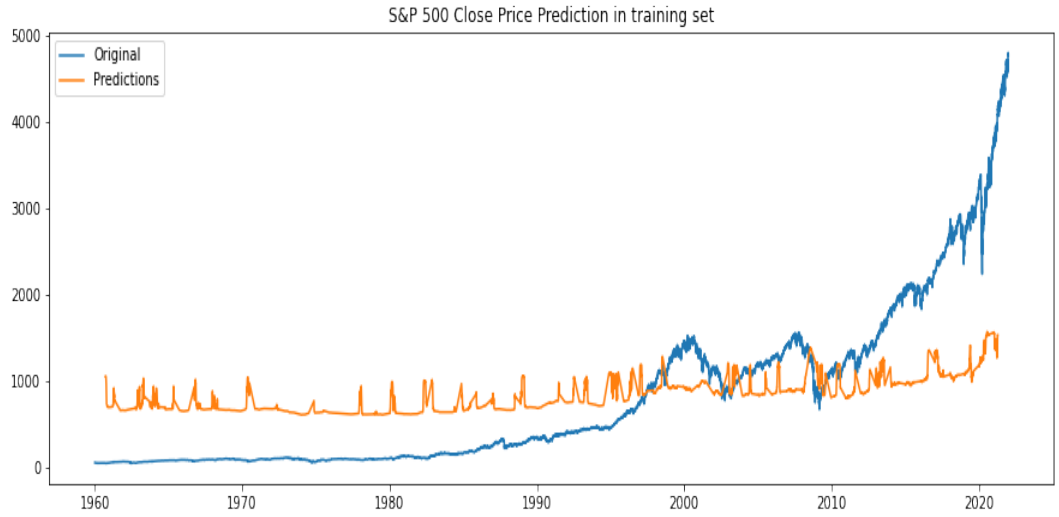

Figure 10: Unoptimized Model Predictions on S&P data

8.2 Predictions Graph on Unoptimized Model from 1960 - 2021

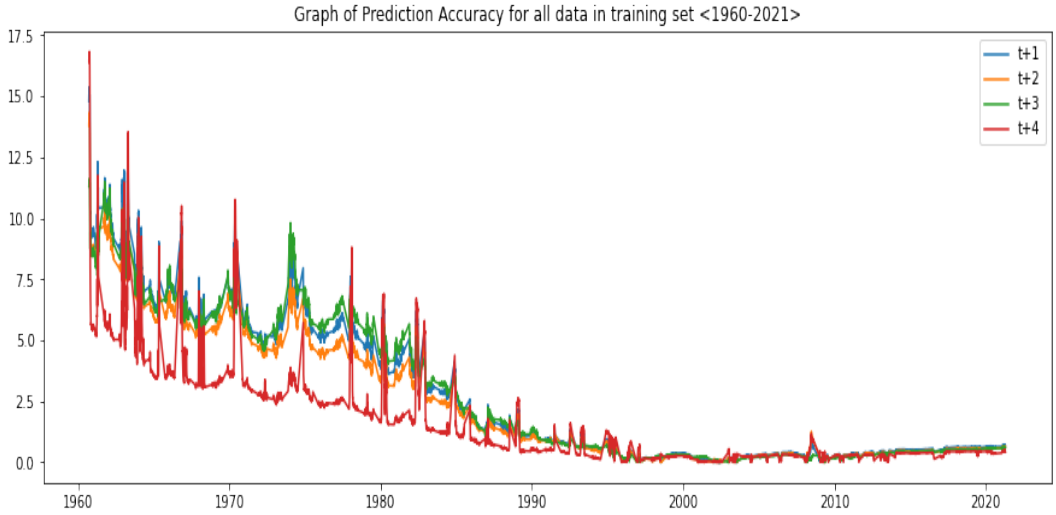

Figure 11: Data Predictions on Unoptimized Model from 1960-2021

8.3 Predictions Graph on Unoptimized Model from 1980 - 2021

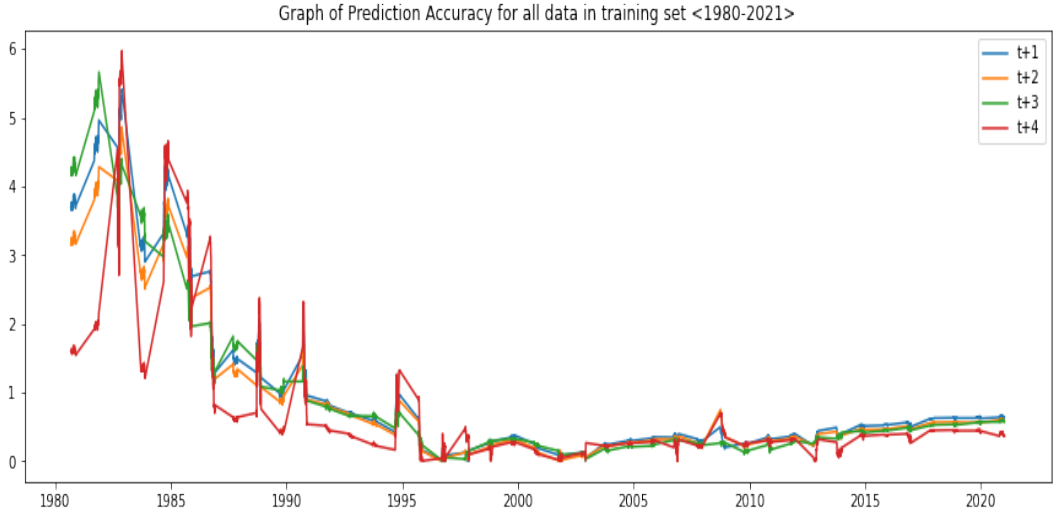

Figure 12: Data Predictions on Unoptimized Model from 1980-2021

8.4 Predictions Graph on Optimized Model from 1960 - 2021

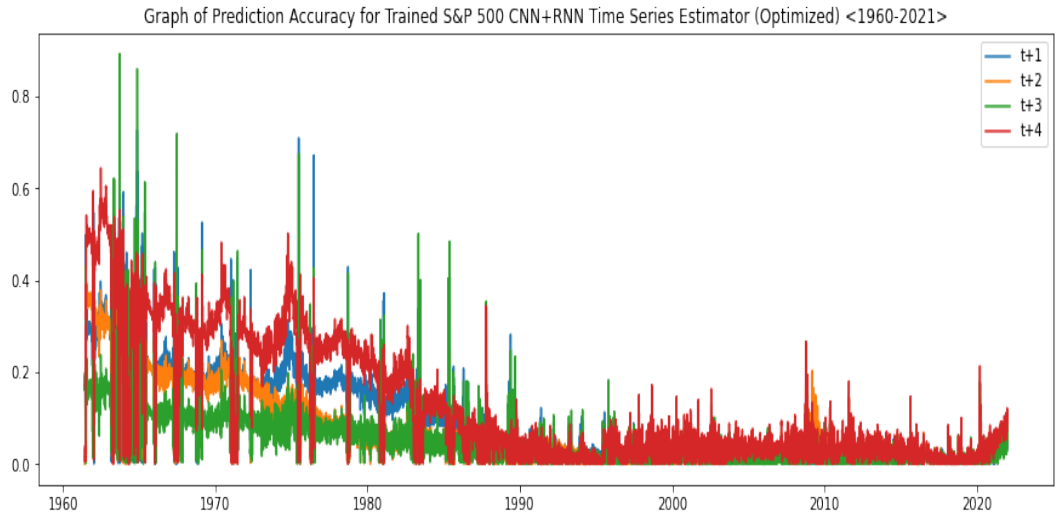

Figure 13: Data Predictions on Optimized Model from 1960-2021

8.5 Predictions Graph on Optimized Model on T+1 days

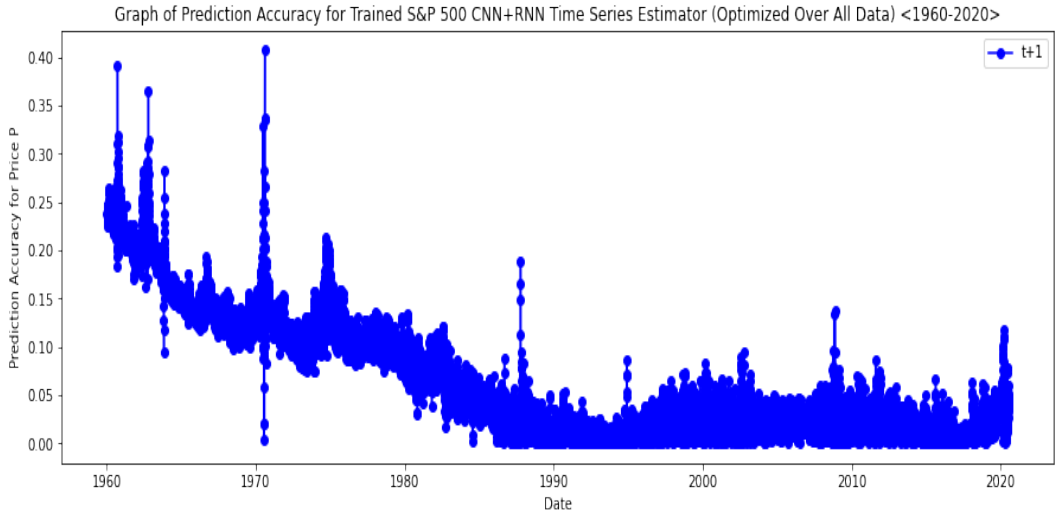

Figure 14: Data Predictions on Optimized Model for T+1 Day

8.6 Predictions Graph on Optimized Model on T+2 days

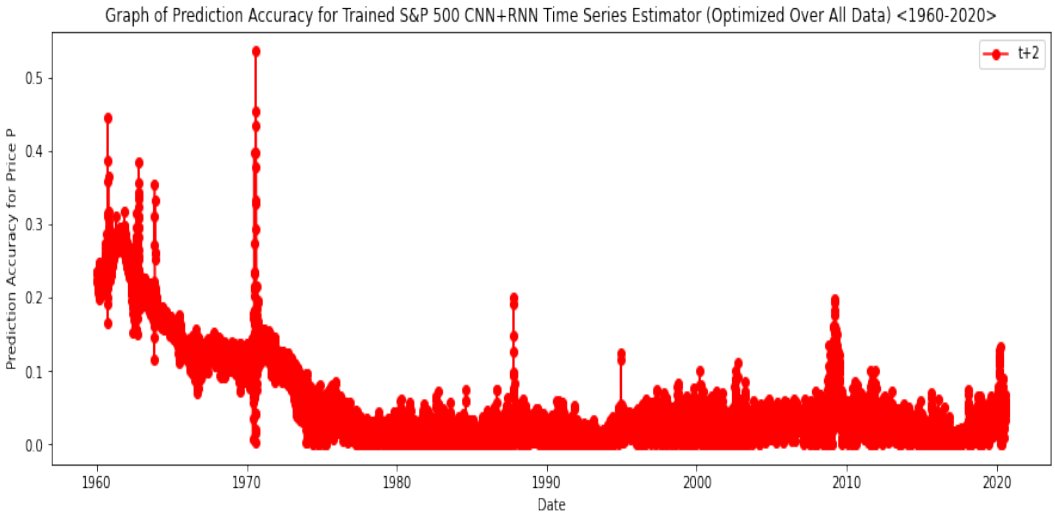

Figure 15: Data Predictions on Optimized Model for T+2 Day

8.7 Predictions Graph on Optimized Model on T+3 days

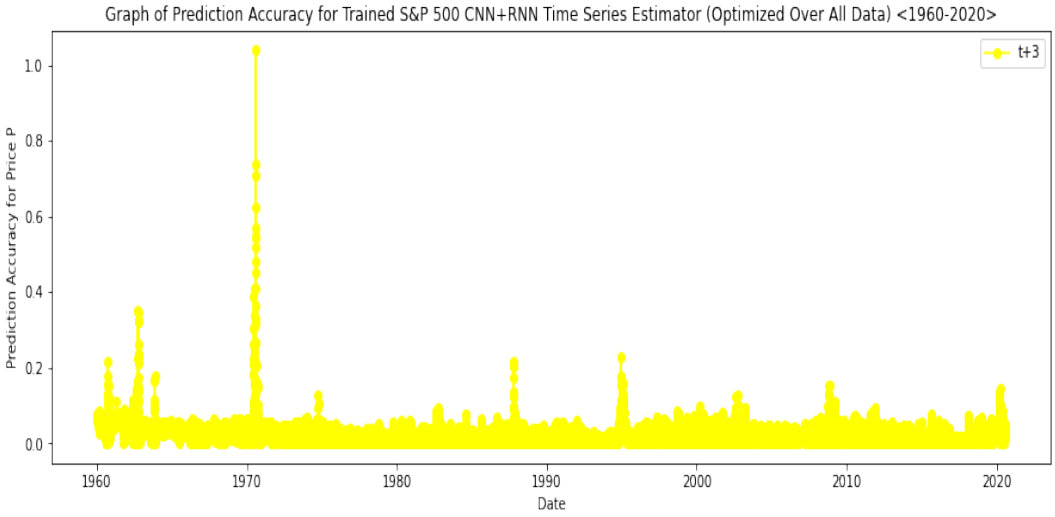

Figure 16: Data Predictions on Optimized Model for T+3 Day

8.8 Predictions Graph on Optimized Model on T+4 days

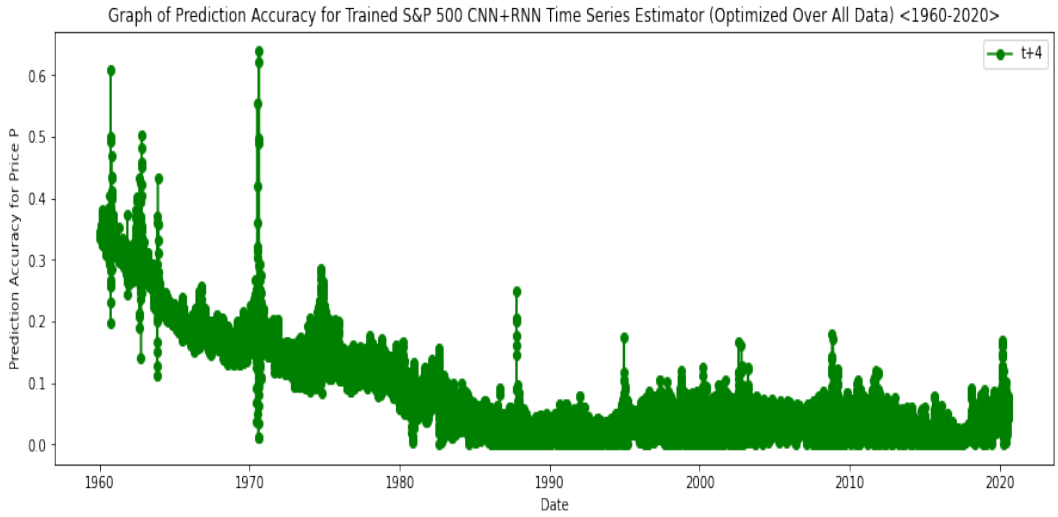

Figure 17: Data Predictions on Optimized Model for T+4 Day

9 Observation of the trading activity for 1 Million Dollars:

The information below shows the trading actions with a 1 million dollar beginning principal. This is a subset of the information. Investment activity is combined with a sell signal, and RNN advises us when the optimum time to buy and sell stocks is.

9.1 Table of trading activity assuming 1 million startup funds:

The data predictions are done in 2 ways where we predicted the data by training model with total data but in other case we just predict from the years after 2000. The money investments during the trading activity is as below.

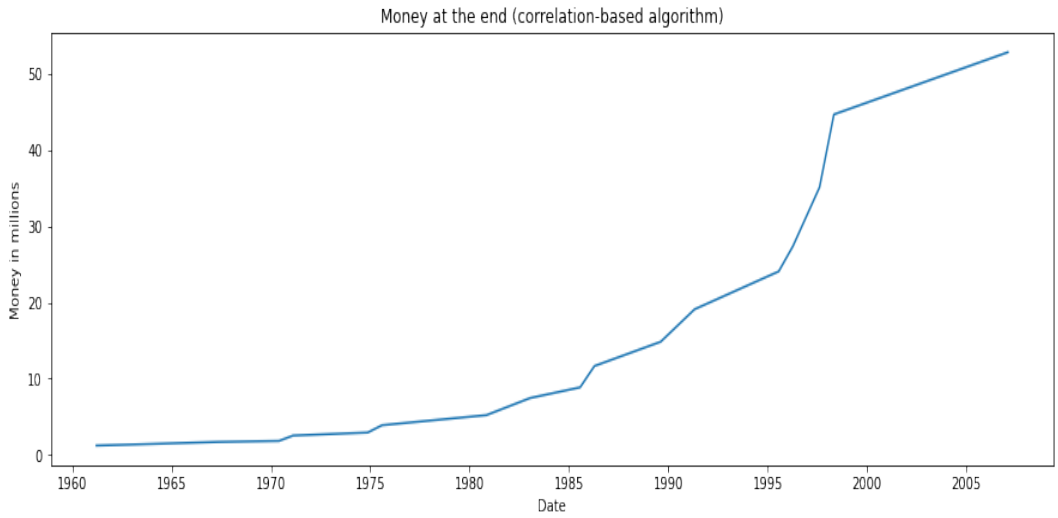

Figure 18: Trading activity for 1 Mil startup funds

10 Introducing noise to input data :

The input data is skewed by the addition of additive gaussian noise. In an excel file, the forecast error has been tabulated. Only a fraction of the table has been displayed in models due to its vast size. Table of outputs for noisy data.

```
def NoiseData(dataset, sigma):
    dataset_noise = np.ndarray(shape=dataset.shape, dtype=np.float32)
    random.seed(1)
    for i in range(len(dataset)):
        s = np.random.normal(0, sigma, 18)
        index = random.sample(list(range(window_size)), 18)
        dataset_noise[i] = dataset[i]
        s = max(X_test[1, index, 0])*s
        temp = dataset_noise[i, index, 0] + s
        temp[temp < 0] = 0.0000001
        dataset_noise[i, index, 0]  = temp
    return dataset_noise
```

Figure 19: Noise addition code

11 Tables of Error Predictions for Noise data:

11.1 Tables of Error Predictions for Noise data for T+1 days:

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1980-05-22	0.164553	0.164552	0.164553	0.164553	0.164553	0.164552	0.164552	0.164553	0.164553	0.164553
1980-05-23	0.173455	0.173455	0.173455	0.173455	0.173455	0.173455	0.173455	0.173456	0.173455	0.173455
1980-05-27	0.169938	0.169951	0.169901	0.169874	0.169953	0.170207	0.170209	0.168333	0.172313	0.164184
1980-05-28	0.162468	0.162466	0.162467	0.162471	0.162475	0.162462	0.162539	0.162374	0.162667	0.163023
1980-05-29	0.138367	0.138336	0.138378	0.138687	0.138632	0.138319	0.137554	0.136376	0.135768	0.142722
...	...	...	...	...	...	...	...	...	...	...
2021-09-21	0.032076	0.032076	0.032076	0.032076	0.032076	0.032076	0.032076	0.032076	0.032076	0.032076
2021-09-22	0.048164	0.048164	0.048165	0.048163	0.048163	0.048169	0.048166	0.048184	0.048196	0.048128
2021-09-23	0.062328	0.062331	0.062329	0.062332	0.062339	0.062311	0.062305	0.062306	0.062411	0.062137
2021-09-24	0.060407	0.060407	0.060407	0.060407	0.060407	0.060407	0.060407	0.060407	0.060407	0.060407
2021-09-27	0.052005	0.052006	0.052005	0.052005	0.052005	0.052005	0.052005	0.052005	0.052005	0.052005

2226 rows × 10 columns

Figure 20: Error Prediction table for noise corrupted T+1 data

11.2 Tables of Error Predictions for Noise data for T+2 days:

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1980-05-23	0.082204	0.082203	0.082204	0.082204	0.082204	0.082204	0.082204	0.082204	0.082205	0.082204
1980-05-24	0.084613	0.084613	0.084613	0.084613	0.084613	0.084613	0.084614	0.084613	0.084614	0.084613
1980-05-28	0.079867	0.079818	0.079824	0.079918	0.079192	0.079055	0.080669	0.078022	0.081315	0.082515
1980-05-29	0.053566	0.053567	0.053558	0.053557	0.053544	0.053577	0.053564	0.053650	0.053676	0.053792
1980-05-30	0.052476	0.052330	0.052698	0.052439	0.052627	0.052200	0.051177	0.048748	0.053658	0.045048
...	...	...	...	...	...	...	...	...	...	...
2021-09-22	0.052454	0.052454	0.052454	0.052454	0.052454	0.052454	0.052454	0.052454	0.052454	0.052454
2021-09-23	0.069966	0.069966	0.069966	0.069966	0.069965	0.069962	0.069968	0.069974	0.069970	0.069969
2021-09-24	0.072607	0.072607	0.072604	0.072604	0.072605	0.072610	0.072584	0.072617	0.072610	0.072547
2021-09-25	0.065152	0.065153	0.065152	0.065152	0.065153	0.065152	0.065153	0.065152	0.065152	0.065153
2021-09-28	0.040465	0.040465	0.040465	0.040465	0.040465	0.040465	0.040465	0.040465	0.040465	0.040465

2226 rows × 10 columns

Figure 21: Error Prediction table for noise corrupted T+2 data

11.3 Tables of Error Predictions for Noise data for T+3 days:

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1980-05-22	0.094674	0.094674	0.094674	0.094674	0.094674	0.094673	0.094674	0.094673	0.094673	0.094674
1980-05-23	0.095829	0.095829	0.095829	0.095829	0.095829	0.095829	0.095830	0.095829	0.095829	0.095829
1980-05-27	0.071346	0.071344	0.071241	0.071285	0.071504	0.072256	0.071244	0.070807	0.073691	0.073384
1980-05-28	0.068953	0.068954	0.068954	0.068953	0.068953	0.068952	0.068947	0.068952	0.068938	0.069001
1980-05-29	0.055100	0.055196	0.055105	0.055069	0.055050	0.055528	0.056079	0.058721	0.054538	0.049857
...	...	...	...	...	...	...	...	...	...	...
2021-09-21	0.047796	0.047796	0.047796	0.047796	0.047796	0.047796	0.047796	0.047796	0.047796	0.047796
2021-09-22	0.055395	0.055395	0.055395	0.055396	0.055395	0.055400	0.055389	0.055397	0.055402	0.055423
2021-09-23	0.054102	0.054100	0.054098	0.054098	0.054095	0.054110	0.054099	0.054088	0.054164	0.054391
2021-09-24	0.029285	0.029285	0.029285	0.029285	0.029285	0.029285	0.029285	0.029285	0.029285	0.029286
2021-09-27	0.025255	0.025255	0.025255	0.025255	0.025255	0.025255	0.025255	0.025255	0.025255	0.025255

2226 rows × 10 columns

Figure 22: Error Prediction table for noise corrupted T+3 data

11.4 Tables of Error Predictions for Noise data for T+4 days:

	0.000	0.001	0.002	0.003	0.005	0.010	0.020	0.030	0.050	0.100
1980-05-22	0.252964	0.252964	0.252963	0.252964	0.252963	0.252963	0.252963	0.252963	0.252964	0.252964
1980-05-23	0.236062	0.236061	0.236061	0.236061	0.236062	0.236062	0.236063	0.236061	0.236061	0.236062
1980-05-27	0.231777	0.231802	0.231662	0.231727	0.232002	0.232151	0.229311	0.230411	0.236447	0.242636
1980-05-28	0.216037	0.216032	0.216036	0.216041	0.216041	0.216110	0.215814	0.215689	0.216196	0.216441
1980-05-29	0.204412	0.204399	0.204431	0.204150	0.204407	0.204424	0.204156	0.204226	0.202332	0.190560
...	...	...	...	...	...	...	...	...	...	...
2021-09-21	0.084557	0.084557	0.084557	0.084557	0.084557	0.084557	0.084557	0.084557	0.084557	0.084557
2021-09-22	0.087755	0.087755	0.087755	0.087754	0.087754	0.087755	0.087751	0.087761	0.087755	0.087764
2021-09-23	0.068902	0.068904	0.068903	0.068904	0.068911	0.068870	0.068892	0.068970	0.068856	0.068939
2021-09-24	0.065254	0.065254	0.065254	0.065254	0.065254	0.065254	0.065254	0.065254	0.065254	0.065254
2021-09-27	0.049158	0.049158	0.049158	0.049158	0.049158	0.049158	0.049158	0.049158	0.049158	0.049158

2226 rows × 10 columns

Figure 23: Error Prediction table for noise corrupted T+4 data

12 Optimized Model Output for Noisy Data:

In this section, we present the details of prediction errors obtained when the dataset was introduced with noise. To ensure legibility and better understanding of the data, each standard deviation has its own table. For the other standard deviations, tables have been prepared in the same way.

12.1 Predictions Graph on Optimized Model for Noisy Data on T+1 days

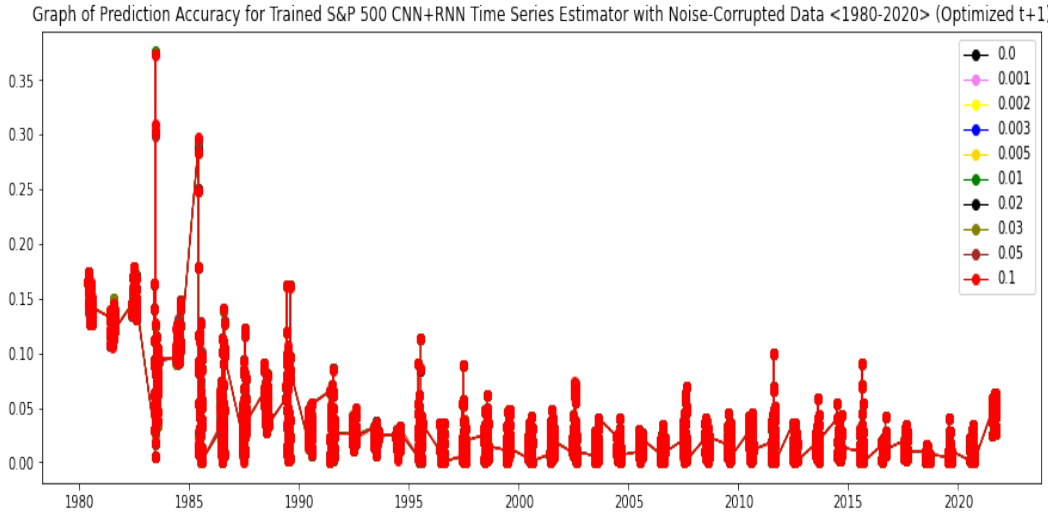

Figure 24: Data Predictions on Noisy Data for T+1 Day

12.2 Predictions Graph on Optimized Model for Noisy Data on T+2 days

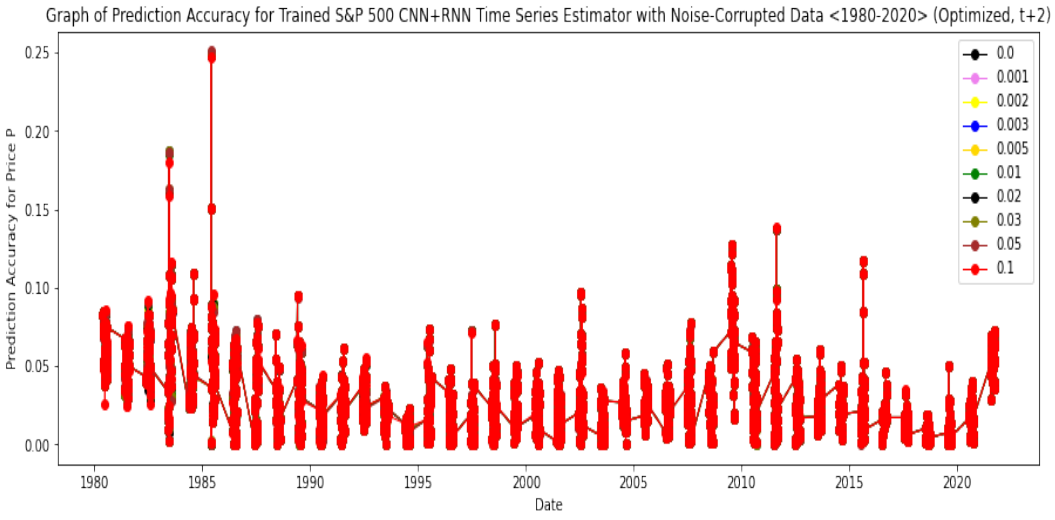

Figure 25: Data Predictions on Noisy Data for T+2 Day

12.3 Predictions Graph on Optimized Model for Noisy Data on T+3 days

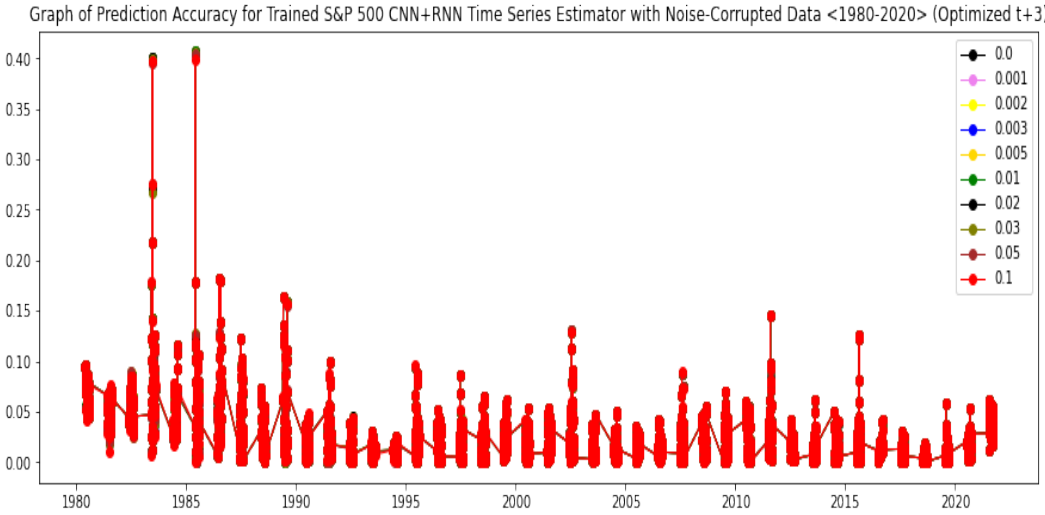

Figure 26: Data Predictions on Noisy Data for T+3 Day

12.4 Predictions Graph on Optimized Model for Noisy Data on T+4 days

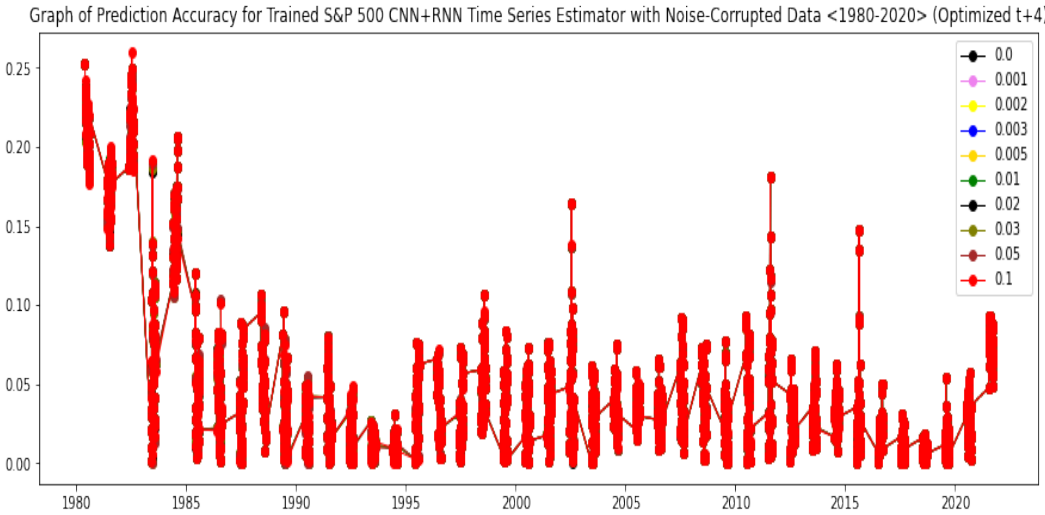

Figure 27: Data Predictions on Noisy Data for T+4 Day

13 Performance Evaluation for the Model:

An attempt was made to use the ReLu activation function, however the model's performance was poor. As a result of the use of the LeakyReLu activation function, the model's performance has significantly improved. The LeakyReLu activation function aids in the avoidance of vanishing gradients. Furthermore, having a mean activation near to zero speeds up the training process.

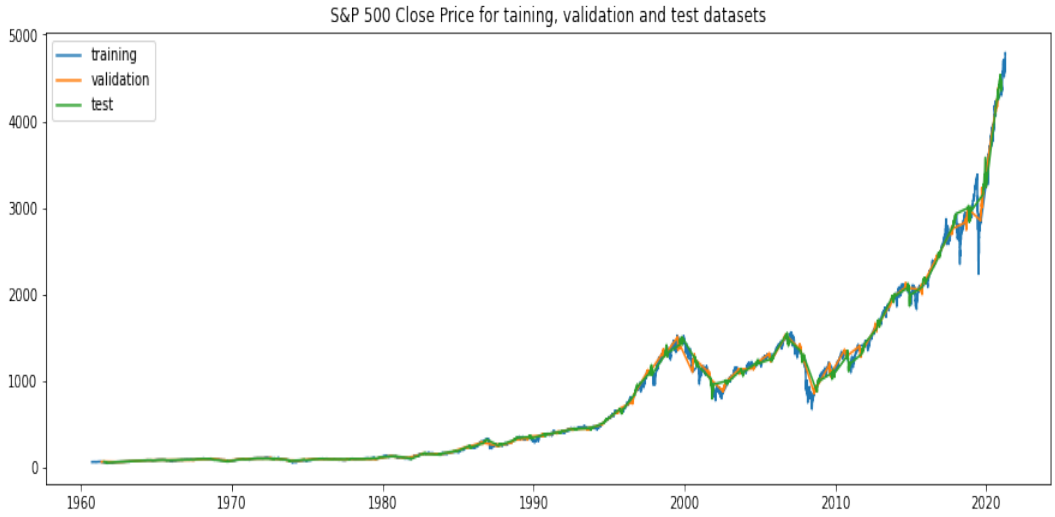

Figure 28: Graph for Train,Test and Validation on data

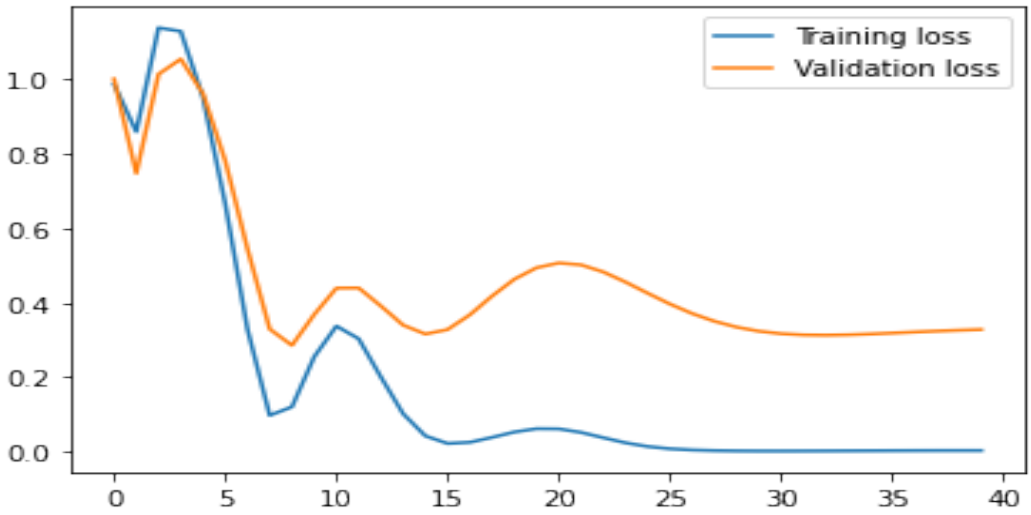

Figure 29: Graph for Training and Validation loss data

14 Discussion:

The overall attitude of investors toward a given securities or financial market is referred to as "market sentiment." It is a market's mood or tone, or its crowd psychology, as reflected by the activity and price movement of the securities traded in that market. In general, rising prices reflect positive market sentiment, whereas dropping prices imply bearish market sentiment. In order to predict buying and selling movements, the model uses market sentiment. The selling probability is given by CNN and used as an input by RNN to improve its performance.

Time series forecasting is about estimating the future value of a time series on the basis of past data. Many time series problems can be solved by looking at a single step in the future. Multi-step time series prediction models the distribution of future values of a signal over a prediction horizon. This approach predicts multiple output values at the same time which is forecasting approach to predict the further course of gradually rising sine wave. In addition, many of these variables are interdependent, which makes statistical. modeling even are more complex.

Because Shiller P/E ratio and adjacent close price are highly correlated, a multivariate RNN model is implemented using the output of CNN, adjacent close price, and Shiller P/E ratio. The selling probability provided as an input to RNN provides more useful information and improves RNN forecasts. By combining the LeakyReLu activation function and the adam optimizer, Optimized CNN+RNN improves the performance of unoptimized CNN+RNN.