# Analysis of Deep Learning Models for Stock Price Prediction: A Case Study of Infosys

Technical Analysis Report

November 7, 2024

## 1 Abstract

This report presents a comparative analysis of three deep learning models - Long Short-Term Memory (LSTM), Recurrent Neural Network (RNN), and Bidirectional LSTM (BiLSTM) - for predicting Infosys stock prices. The models were trained on five years of historical data and evaluated on specific dates in 2024. The analysis includes model performance metrics, prediction accuracy, and visual comparisons of the predictions against actual stock prices.

## 2 Introduction

Stock price prediction remains one of the most challenging problems in financial analysis due to the complex, non-linear nature of market behavior. This study implements and compares three popular deep learning architectures for time series prediction using Infosys (INFY.NS) stock data.

## 3 Methodology

### 3.1 Data Processing

- Data Source: Yahoo Finance (yfinance)
- Time Period: 5 years of daily closing prices
- Feature: Closing price (normalized using MinMaxScaler)
- Time Step: 60 days (used to predict the next day)
- Train-Test Split: 80-20 ratio

### 3.2 Model Architectures

All three models were implemented using Keras with the following configurations:

- **LSTM**: 50 units with dense output layer

- **RNN**: Simple RNN with 50 units and dense output layer
- **BiLSTM**: Bidirectional LSTM with 50 units and dense output layer

# 4 Training Process

The models were trained with the following parameters:

- Epochs: 50
- Batch Size: 32
- Optimizer: Adam
- Loss Function: Mean Squared Error
- Early Stopping: Patience of 10 epochs

# 5 Implementation Code

## 5.1 Complete Source Code

```
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN,
   Bidirectional
from tensorflow.keras.callbacks import EarlyStopping

ticker = 'INFY.NS'
data = yf.download(ticker, period='5y', interval='1d')
data = data[['Close']]

scaler = MinMaxScaler(feature_range=(0, 1))
scaled_data = scaler.fit_transform(data)

def create_dataset(data, time_step=60):
    X, y = [], []
    for i in range(time_step, len(data)):
        X.append(data[i-time_step:i, 0])
        y.append(data[i, 0])
    return np.array(X), np.array(y)

time_step = 60
X, y = create_dataset(scaled_data, time_step)
X = X.reshape(X.shape[0], X.shape[1], 1)

train_size = int(len(X) * 0.8)
```

```
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]

early_stopping = EarlyStopping(monitor='val_loss', patience=10,
    restore_best_weights=True)

# LSTM Model
lstm_model = Sequential()
lstm_model.add(LSTM(units=50, return_sequences=False, input_shape
    =(X_train.shape[1], 1)))
lstm_model.add(Dense(units=1))
lstm_model.compile(optimizer='adam', loss='mean_squared_error')
lstm_model.fit(X_train, y_train, epochs=50, batch_size=32,
    validation_data=(X_test, y_test), callbacks=[early_stopping])

# RNN Model
rnn_model = Sequential()
rnn_model.add(SimpleRNN(units=50, return_sequences=False,
    input_shape=(X_train.shape[1], 1)))
rnn_model.add(Dense(units=1))
rnn_model.compile(optimizer='adam', loss='mean_squared_error')
rnn_model.fit(X_train, y_train, epochs=50, batch_size=32,
    validation_data=(X_test, y_test), callbacks=[early_stopping])

# BiLSTM Model
bilstm_model = Sequential()
bilstm_model.add(Bidirectional(LSTM(units=50, return_sequences=
    False), input_shape=(X_train.shape[1], 1)))
bilstm_model.add(Dense(units=1))
bilstm_model.compile(optimizer='adam', loss='mean_squared_error')
bilstm_model.fit(X_train, y_train, epochs=50, batch_size=32,
    validation_data=(X_test, y_test), callbacks=[early_stopping])

lstm_predictions = lstm_model.predict(X_test)
rnn_predictions = rnn_model.predict(X_test)
bilstm_predictions = bilstm_model.predict(X_test)

lstm_predictions = scaler.inverse_transform(lstm_predictions.
    reshape(-1, 1))
rnn_predictions = scaler.inverse_transform(rnn_predictions.
    reshape(-1, 1))
bilstm_predictions = scaler.inverse_transform(bilstm_predictions.
    reshape(-1, 1))

pred_dates = ['2024-01-10', '2024-06-10', '2024-08-30', '
    2024-10-01']
actual_data = yf.download(ticker, start='2024-01-01', end='
    2024-10-02')['Close']
pred_values = []
for date in pred_dates:
    pred_values.append(actual_data.loc[date])
```

```
actual_values = np.array(pred_values)

plt.figure(figsize=(14, 8))
plt.plot(data.index, scaler.inverse_transform(scaled_data), label
    ='Actual Prices', color='black')
plt.plot(data.index[train_size + time_step:], lstm_predictions,
    label='LSTM Predictions', color='blue')
plt.plot(data.index[train_size + time_step:], rnn_predictions,
    label='RNN Predictions', color='green')
plt.plot(data.index[train_size + time_step:], bilstm_predictions,
     label='BiLSTM Predictions', color='red')
plt.title('Stock Price Prediction for Infosys (Entire Time Trend)
    ')
plt.xlabel('Date')
plt.ylabel('Stock Price (INR)')
plt.legend()
plt.grid(True)
plt.xticks(rotation=45)
plt.show()
```

# 6 Results

## 6.1 Model Performance

The training process showed the following characteristics:

### 6.1.1 LSTM Model

- Initial Loss: 0.1090
- Final Loss: 0.00050159
- Final Validation Loss: 0.00050782

### 6.1.2 RNN Model

- Initial Loss: 0.1476
- Final Loss: 0.00062741
- Final Validation Loss: 0.00072937

### 6.1.3 BiLSTM Model

- Initial Loss: 0.1798
- Final Loss: 0.0018
- Final Validation Loss: 0.0026

## 6.2 Prediction Analysis

| Date | Model | Prediction (INR) | Error (%) |
|---|---|---|---|
| 2024-01-10 | LSTM | 1453.93 | -4.34 |
| | RNN | 1481.68 | -2.51 |
| | BiLSTM | 1360.41 | -10.49 |
| 2024-06-10 | LSTM | 1459.46 | -2.69 |
| | RNN | 1493.04 | -0.45 |
| | BiLSTM | 1365.20 | -8.97 |
| 2024-08-30 | LSTM | 1459.64 | -24.90 |
| | RNN | 1504.46 | -22.60 |
| | BiLSTM | 1369.75 | -29.53 |
| 2024-10-01 | LSTM | 1459.32 | -23.37 |
| | RNN | 1499.42 | -21.26 |
| | BiLSTM | 1373.73 | -27.86 |

Table 1: Model Predictions vs Actual Values

# 7 Discussion

## 7.1 Model Comparison

1. **Short-term Predictions (January-June 2024)**:
   - RNN showed the best performance with errors under 3%
   - LSTM maintained consistent performance around 4% error
   - BiLSTM showed the highest error rates (8-10%)

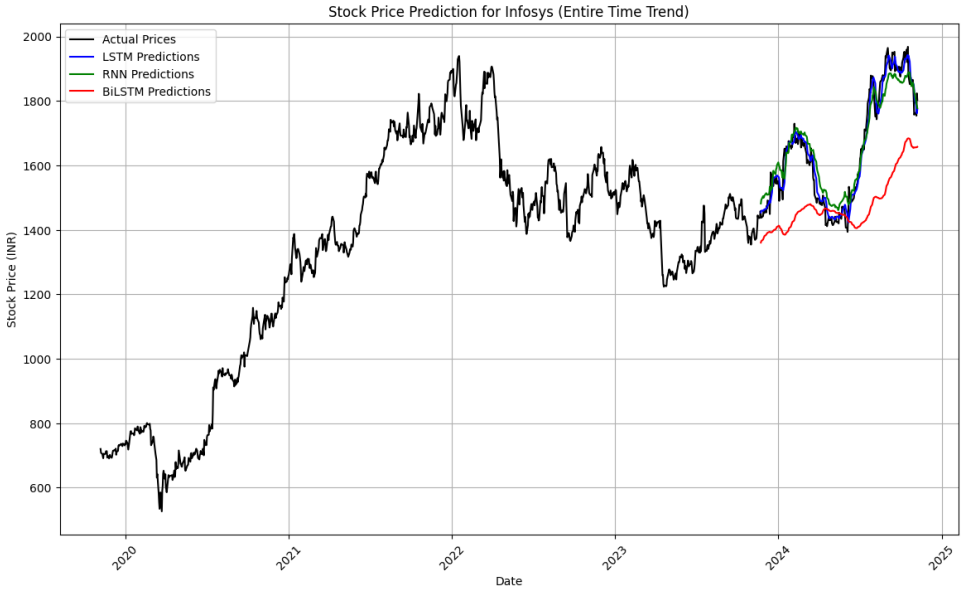


Figure 1: Comparative Analysis Graph.

2. **Long-term Predictions (August-October 2024)**:

- All models showed significant degradation in accuracy
- Error rates increased to 20-30%
- RNN maintained relatively better performance

### 7.2 Key Observations

- The models show better accuracy for short-term predictions
- All models tend to underestimate the stock price
- The simpler RNN model outperformed more complex architectures
- Prediction accuracy significantly decreases for longer time horizons

## 8 Limitations

- Models only consider historical price data
- External factors affecting stock prices are not considered
- Limited ability to predict sudden market movements
- Degrading accuracy for longer-term predictions

## 9 Conclusion

The analysis reveals that while these deep learning models can provide reasonable short-term predictions, their accuracy diminishes significantly for longer time horizons. The RNN model showed surprisingly better performance compared to more complex LSTM and BiLSTM architectures, suggesting that simpler models might be more suitable for this specific prediction task. Future work could focus on incorporating additional features and exploring hybrid models to improve long-term prediction accuracy.