

LSTM Prediction of Foreign Exchange Rates (Individual Report)

1. Introduction:

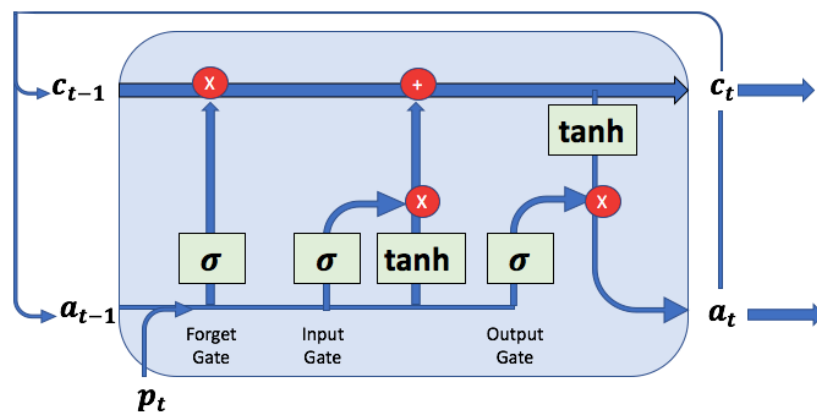
This study uses Long-Short Term Memory neural networks for time series prediction of the Brazilian Real/United States Dollar (BRL/USD) and Swiss Franc (CHF/USD) foreign exchange rates. My partner and I tested the LSTM's forecast accuracy on both the level and differenced day to day movements of both currencies.

2. Scope of Individual Work and Model Background

I contributed to all phases of the project completion and neural network design. My contributions were most heavily focused on data preprocessing, designing the approach, and writing/running the code for the three neural network experiments (described below).

The model architecture we use for prediction is the Long Short-Term Memory network that was first proposed in 1997. The only modification we make to the LSTM cell included in the model architecture is the addition of a forget gate, which has since become standard practice in LSTM forecasting (it is even included in the default specification in KERAS).¹ A diagram of the LSTM network that I made and adapted from our research is shown below. A full write up of the equations included in the LSTM network is included in the Group Final paper.

Figure 1: LSTM Cell Diagram



¹Sepp Hochreiter and Jurgen Schmidhuber, "Long Short-Term Memory," Neural Computation (1997): 1735-1780.

LSTM's use sequential data to create predictions of future values using a specified lookback. In the LSTM, the previous forecast value and the inputs go through a series of gates to determine the next value in the series. In the forget gate, the model determines how much of the previous forecast and inputs to remember when making a new prediction. Next, the input gate determines which values in the LSTM cell to update. Finally, the output gate determines how much of the inputs to write the cell's state. LSTMs are organized in this way to impose a constant error flow. This helps to solve for the vanishing gradient problem, where weights and biases decay exponentially towards zero in the multiplicative operations used in dynamic backpropagation.²

Our model connects an LSTM cell to a dropout layer to prevent overfitting by probabilistically removing neurons at each iteration. After dropout, the model is connected to a final, densely connected layer that converts the multidimensional output of the LSTM cell to a scalar forecast. The code for the basic model architecture included in the paper is shown below.

```
model = Sequential()
model.add(LSTM(cells[i], activation='tanh', input_shape=(2,1)))
model.add(Dropout(rate = 0.1))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

3. Detailed Scope of Work

3.1 Preprocessing and Exploratory Data Analysis

In the preprocessing and exploratory data analysis phase I downloaded both daily time series from the Federal Reserve Economic Database (FRED) and put them into a data frame. I also used scikit learn's MinMax Scaler function to normalize each time series to be between values of 0 and 1 in order to make the training process more stable. Finally, I used pandas' built in interpolate function to linearly interpolate missing values in the dataset. All of these results can be found in the data_clean_edu.py file.

For exploratory data analysis, I focused on testing for stationarity and identified the autoregressive components to include in the model's lookback period. The LSTM lookback consists of how many previous values to include in the input vector. In traditional time series economics, this is similar to determining how many autoregressive lags (i.e. values of the variable in the past) to include as predictors in the forecast. To determine this number, I used python's statsmodels package to create plots of each currency's autocorrelation and partial autocorrelation. For each specification, I set the lookback period to the number of statistically significant lags observed in the PACF charts. I found the CHF/USD series to be an autoregressive

²Colah, "Understanding LSTM Networks," Colah.github.com, August 27, 2015. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

process with two significant lags and the BRL/USD process to be an autoregressive process with only one significant lag. I also found both series to be non-stationary and integrated to the first order by using statsmodel's implementation of the augmented dickey fuller test. I found that differences in the daily foreign exchange rate levels for both series were weakly stationary. A full write up of these tests is provided in the group paper.

My final contribution in preprocessing was writing and adapting two functions to shape and index our data for the training and testing process. First, I adapted a similar function created by Jason Brownlee of *Machine Learning Mastery* to reshape the datasets into the proper input shape of an LSTM, (number of samples, time steps, input dimensionality). I did not copy this code directly, but used it as a reference to develop my own method of slicing the provided time series using pandas and numpy. The function first creates a two dimensional dataframe where each column is a shifted lag of the variable at point in time $t-n$, up to a specified lag length. It then converts the dataframe to a numpy array and adds the third dimension, with a defaulted value of one, to reshape inputs into an array called X. It also outputs a y array that is just the series at point in time t .³

The second function I created was `train_test_idx`, which creates a list of indexes to slice the X and y datasets with. It is built around scikitlearn's TimeSeries Split function and outputs two lists of train and test sets. Each element in the train and test lists is its own list. In the train list, each list contains an increasing number of sequential years to train the data on. For example, `train[0]` is the first year in our dataset, while `train[1]` is the first and second, and so on. The test list contains lists of the holdout year values immediately after the training period. So for instance, `test[0]` contains the second year of dataset and `test[1]` contains the third. We used these slices to conveniently call slices of the X and y arrays to configure different training and testing periods.

3.2 Training and Testing (Experiment # 1)

I designed our process for training and testing the optimal configuration of neurons in the LSTM cell and ran code for both the CHF/USD series and the BRL/USD series. For the base case in our analysis, we trained on the first three years of data and tested on the final year of the data. To determine the correct number of neurons to include in the LSTM cell, I ran a simple for loop for each time series that tested the performance of neuron configurations between 10 and 100, increasing by 10. I then compared results and picked the configuration with the lowest mean squared error (MSE). In cases where the MSE would plateau after certain instances or if a few configurations would have approximately the same MSE, I picked the configuration with the lowest number of neurons.

³ Jason Brownlee, "How to Develop LSTM Networks for Time Series Forecasting," *Machine Learning Mastery*, November 14, 2018. <https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>

After determining the optimal neuron configuration, I re-initialized the model and tested again only at the optimal number of neurons and then analyzed the results. First, I plotted the loss functions of the train and test sets generated by the model using Keras' history feature. In all specifications, I did not observe an increase in the validation error that would have indicated overfitting. Second, I plotted the autocorrelation function and tested for stationarity on the model residuals. This was to ensure that errors of the model were not serially correlated, which would bias forecast estimates. Finally, I created plots of the predicted versus actual results of our forecast.

3.3 Validation and Back testing (Experiment # 2 and Experiment # 3)

I also wrote and ran all of the code for our validation approach used in our project in consultation with our professor. We did two types of validation for each time series included in the analysis. In experiment # 2, I re-initialized and trained the model on only the first two years of the data (instead of the first three in the base specification), and then tested on the third year (instead of the fourth year). I then compared plots of the model loss, autocorrelation of the residuals, forecasts, and mean square errors to what was generated in the base forecast. I considered it a positive indicator of generalizability if results were consistent in experiment # 1 and experiment # 2.

In experiment # 3, I initialized the model and trained and tested it again using a back testing approach. This time I trained on the third year and tested on the second year of the dataset. This ensured another test of stability in our model results. Again, I compared the model loss plots, autocorrelation of residuals, predicted versus actual values and mean square errors to determine consistency.

4. Results

For each series other than the differenced CHF/USD rate, I found that 20 neurons produced an optimal forecast estimate. In both the raw CHF/USD rates and BRL/USD rates, I observed a diminishing impact in reducing the MSE relative to an increase in the number of LSTM neurons above 20. Accordingly, I determined that this point was when model performance was deemed adequate enough to proceed.

In the differenced CHF/USD rate, I did not notice any difference between model performance in LSTM configurations with more than 10 neurons, and observed adequate performances below 10 neurons as well. As such, I chose a five neuron configuration to reduce the risk of overfitting while still allowing for some degree of complexity in the learning process. Differenced BRL/USD rates had the most interesting results and exhibited high variability in model performance at different neuron configurations. LSTM cell configurations with 20, 50, 70, and 100 neurons produced the lowest MSE in the training period. I chose to utilize 20 neurons as it was the simplest configuration of the model with the lowest error. The motivation for this decision was that a simple model would reduce the risk of overfitting.

Figure 2: LSTM neuron configurations and losses: CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right).

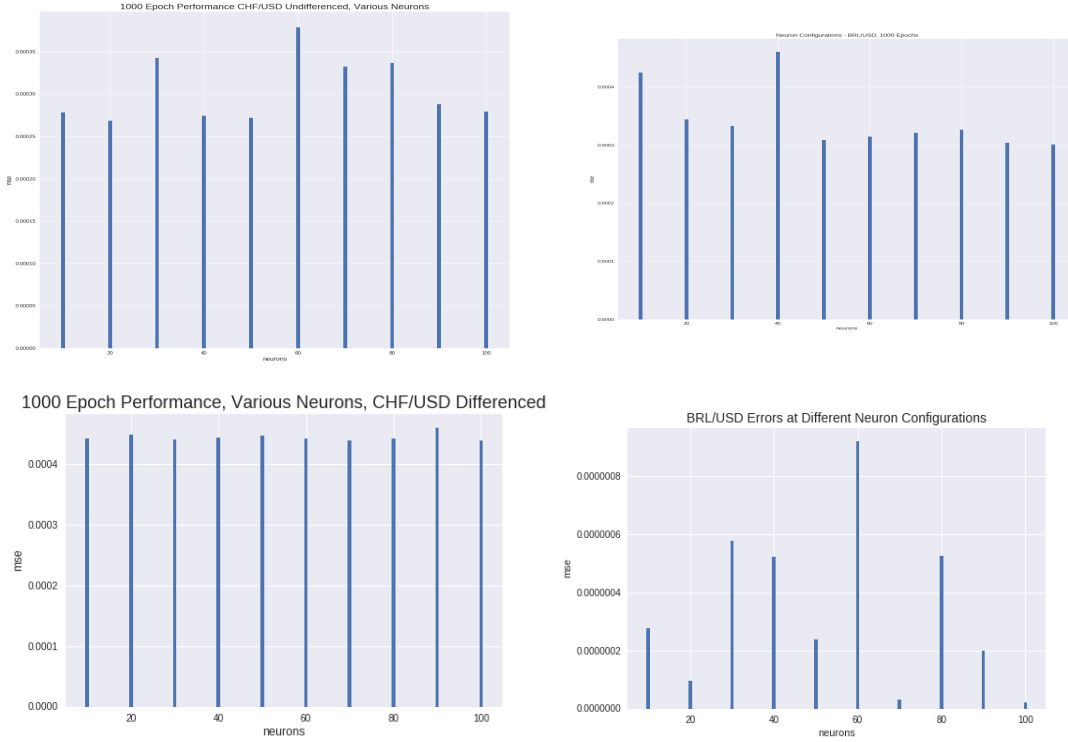


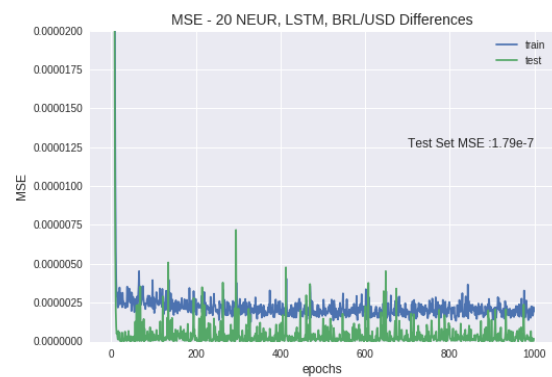
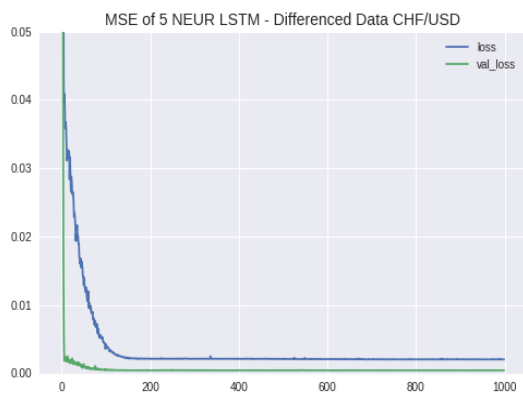
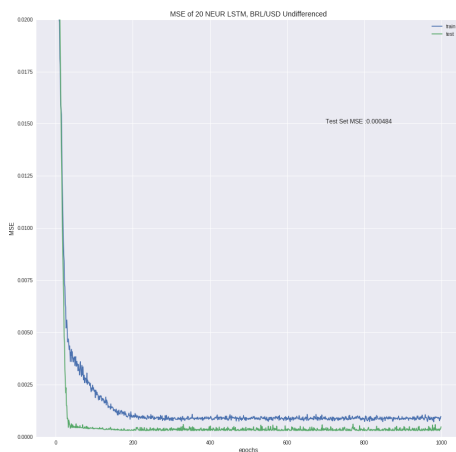
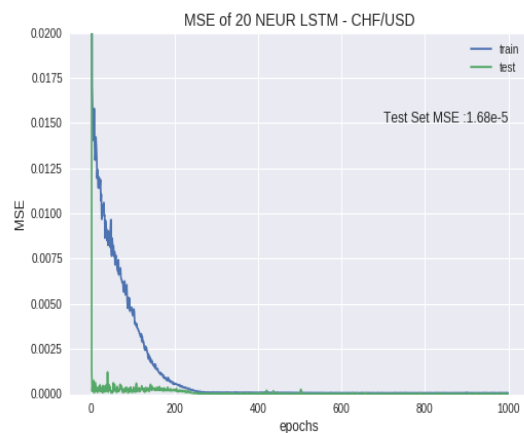
Table 1: Optimal Number of Neurons Included in Forecast LSTM

CHF/USD	BRL/USD	CHF/USD Differences	BRL/USD Differences
20	20	5	20

4.1 Training and Testing the LSTM specifications (Experiment # 1)

After determining the optimal number of neurons to include in the LSTM cell, I trained each specification on the first three years of data and tested its accuracy on the final year. Plots of the model loss in the training and holdout sets in each specification shown an asymptotic decay to zero around 200 epochs. In all specifications, the loss function on the holdout year stays level and does not trend upwards after a certain number of epochs in the training process. Increases in the validation error would indicate poor generalizability on data not used in training. Since the plots of the model loss remain consistent between the training and testing sets, it appears that the models are not overfitting.

Figure 3: Model Loss Plots: CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right). Training set shown in blue, Testing set shown in green.



To test for serially correlated errors in predictions, I plotted the autocorrelation functions of the model residuals between predictions and actual values in the test set. The ACF of the model residuals show a few statistically significant lags after an impulse at zero. However, I found that the residuals are weakly stationary after conducting ADF tests in each time series. These stationary residuals suggest that errors are not serially correlated and that forecast estimates made by each model are unbiased.

Figure 4: ACF Plots of Forecast Residuals: CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right).

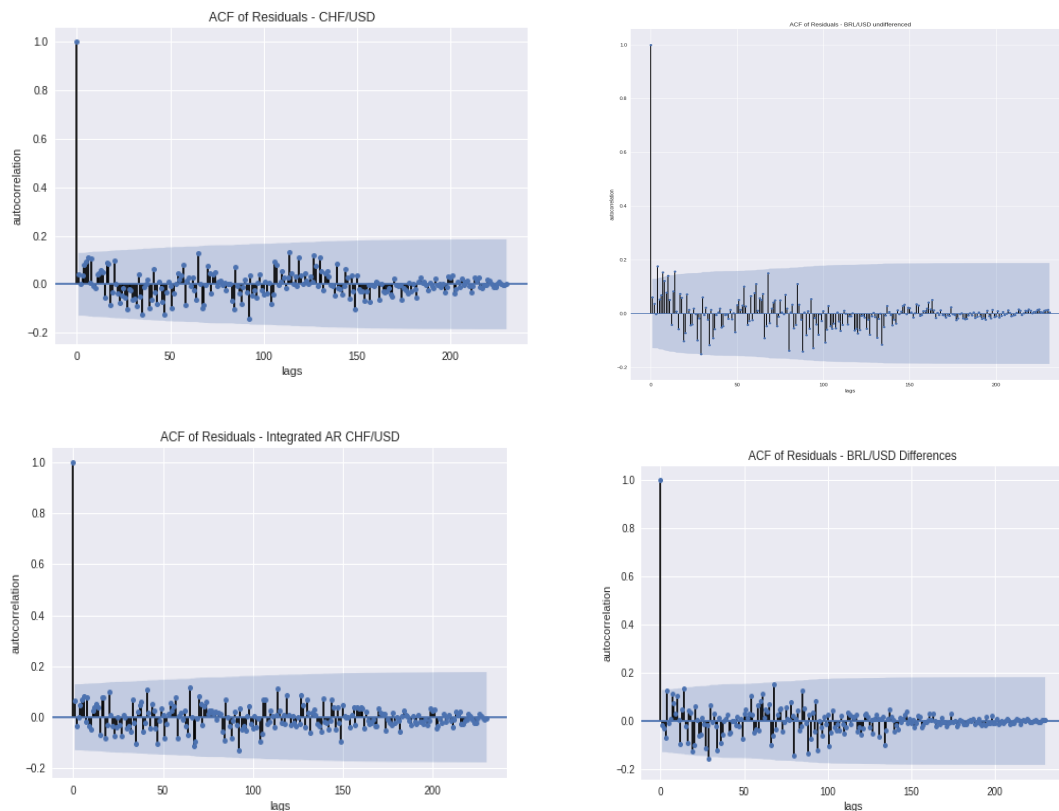


Table 2: ADF Test Statistics in Levels and First Differences. *, **, *** indicates respective 10%, 5%, and 1% confidence

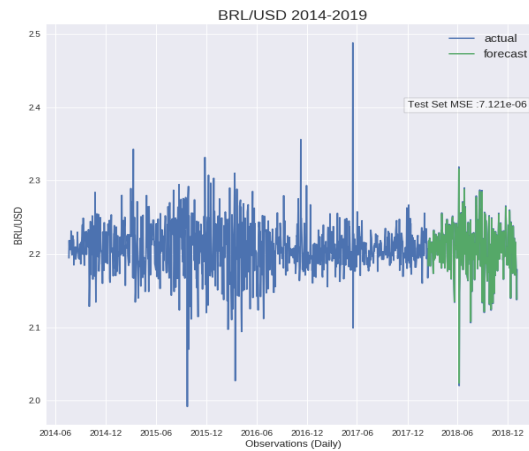
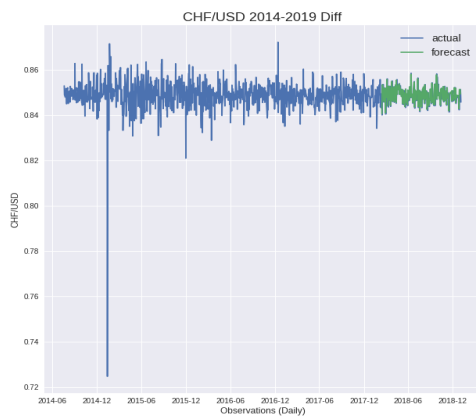
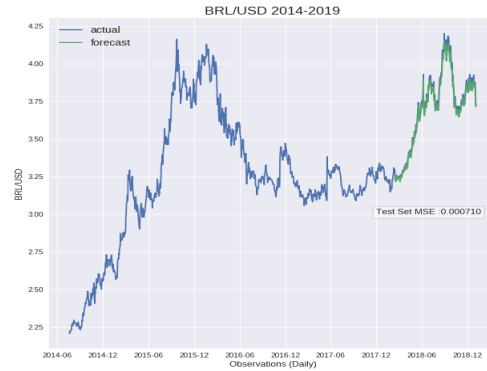
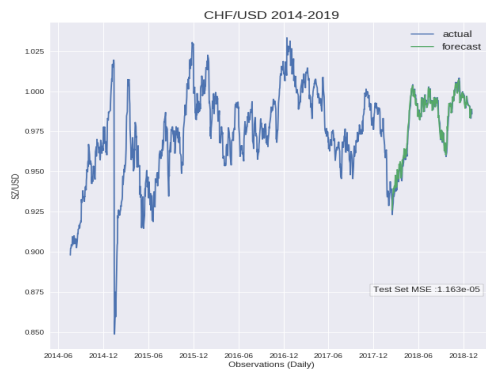
Time Series	Levels	First Differences
CHF/USD	-14.32***	-13.803***
BRL/USD	14.55***	2.691*

Finally, I plotted the predicted and actual values of each time series and compared their mean square errors. Visual inspection suggests that the model does a good job of approximating movement in both the raw and the differenced daily foreign exchange rate movements. The forecast MSE in each specification is also quite low, which suggests goodness of fit and forecast accuracy.

Table 3: Forecasting Mean Squared Errors

Time Series	Levels	First Differences
CHF/USD	$1.2 * 10^{-5}$	$1.723 * 10^{-5}$
BRL/USD	$7.1 * 10^{-4}$	$7.1 * 10^{-6}$

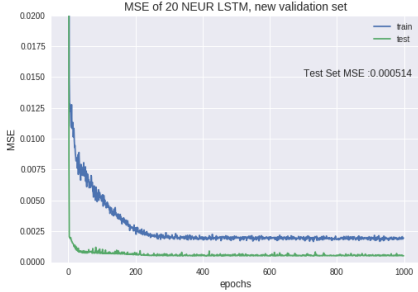
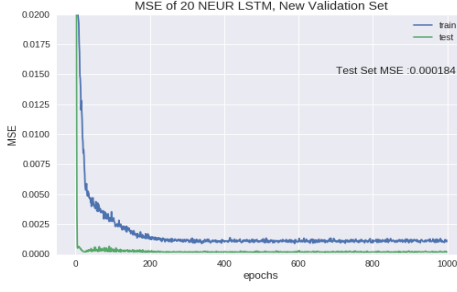
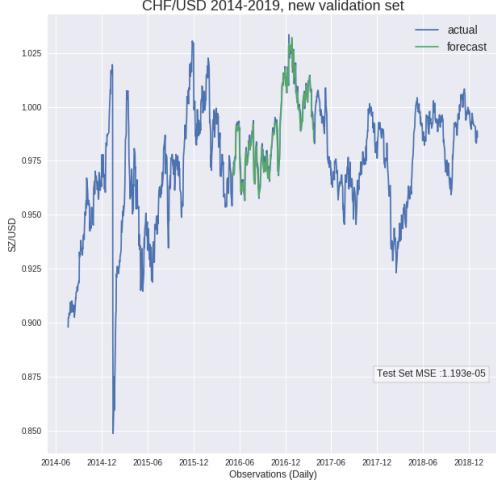


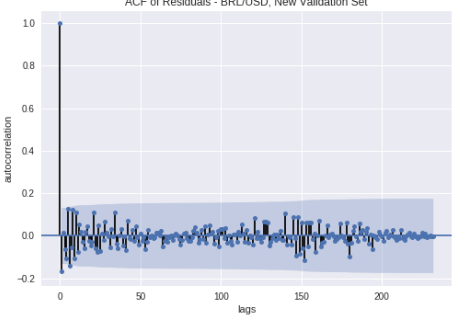
Figure 5: Forecasting Results: CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right). Predicted values are green and actual values are blue.



4.2 Validation (Experiment # 2)

Validating our model on new subsamples of the dataset supported our findings that the LSTM forecasts we conducted are accurate. Here, I trained on the first two years and tested on the third year of the data instead of training on the first three years and testing on the fourth year like I did in the original forecasts. Each specification produces consistent estimates to the ones presented in experiment # 1. Model loss plots show an asymptotic descent to zero in both the train and validation sets, and ACF plots of the prediction residuals are shown to be weakly stationary. This suggests again that our model generalizes well and produces unbiased estimates. Moreover, the predicted versus actual plots of the forecasts show a good fit to the data and low mean square error in both the level and first differenced CHF/USD and BRL/USD time series.

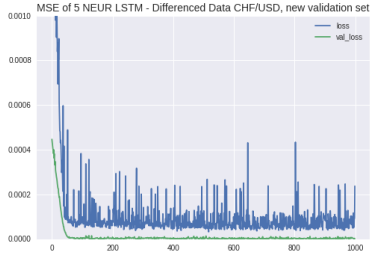
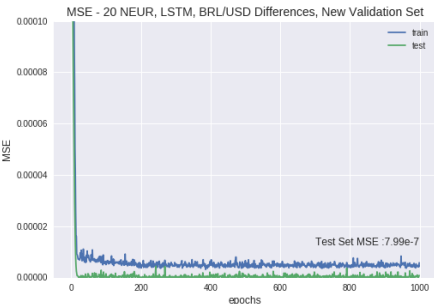
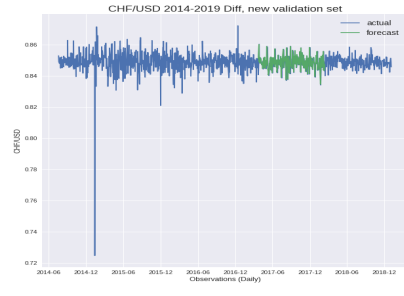
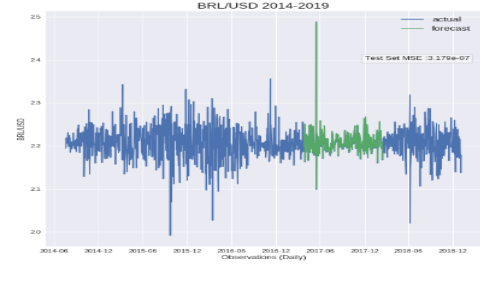
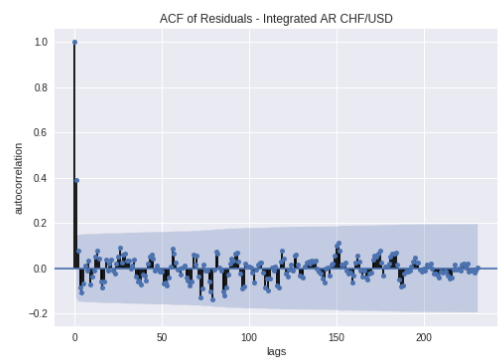
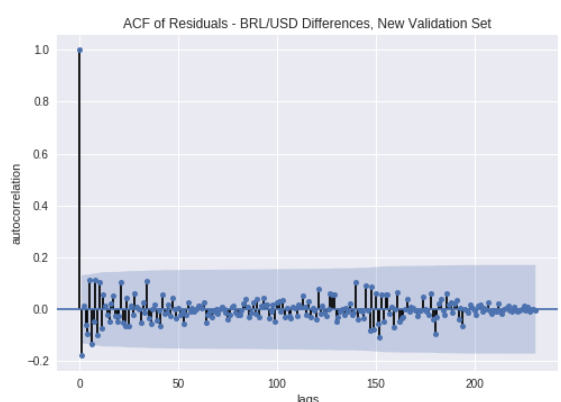
Experiment # 2	
CHF/USD	BRL/USD

	
MSE	MSE
	
FORECASTING	FORECASTING
	
RESIDUAL ACF	RESIDUAL ACF

Experiment # 2

CHF/USD First Differences

BRL/USD First Differences

	
MSE	MSE
	
	FORECASTING
	
Residual ACF	RESIDUAL ACF

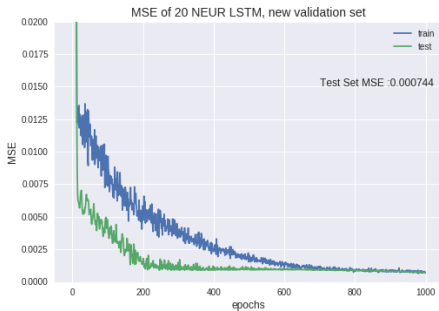
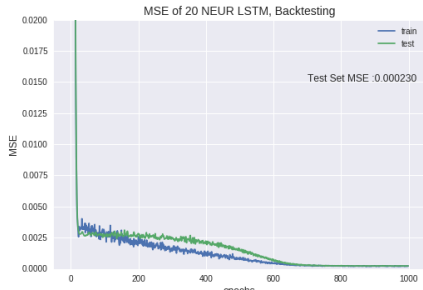
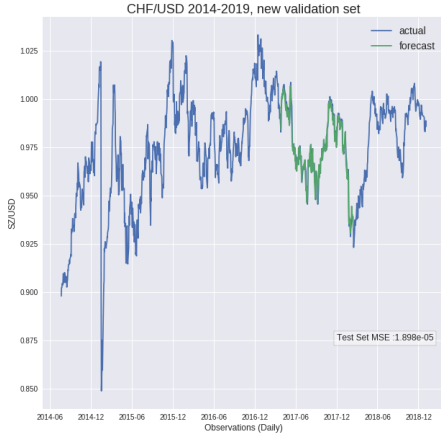
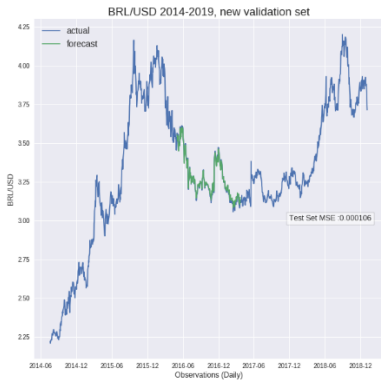
4.3 Back testing (Experiment # 3)

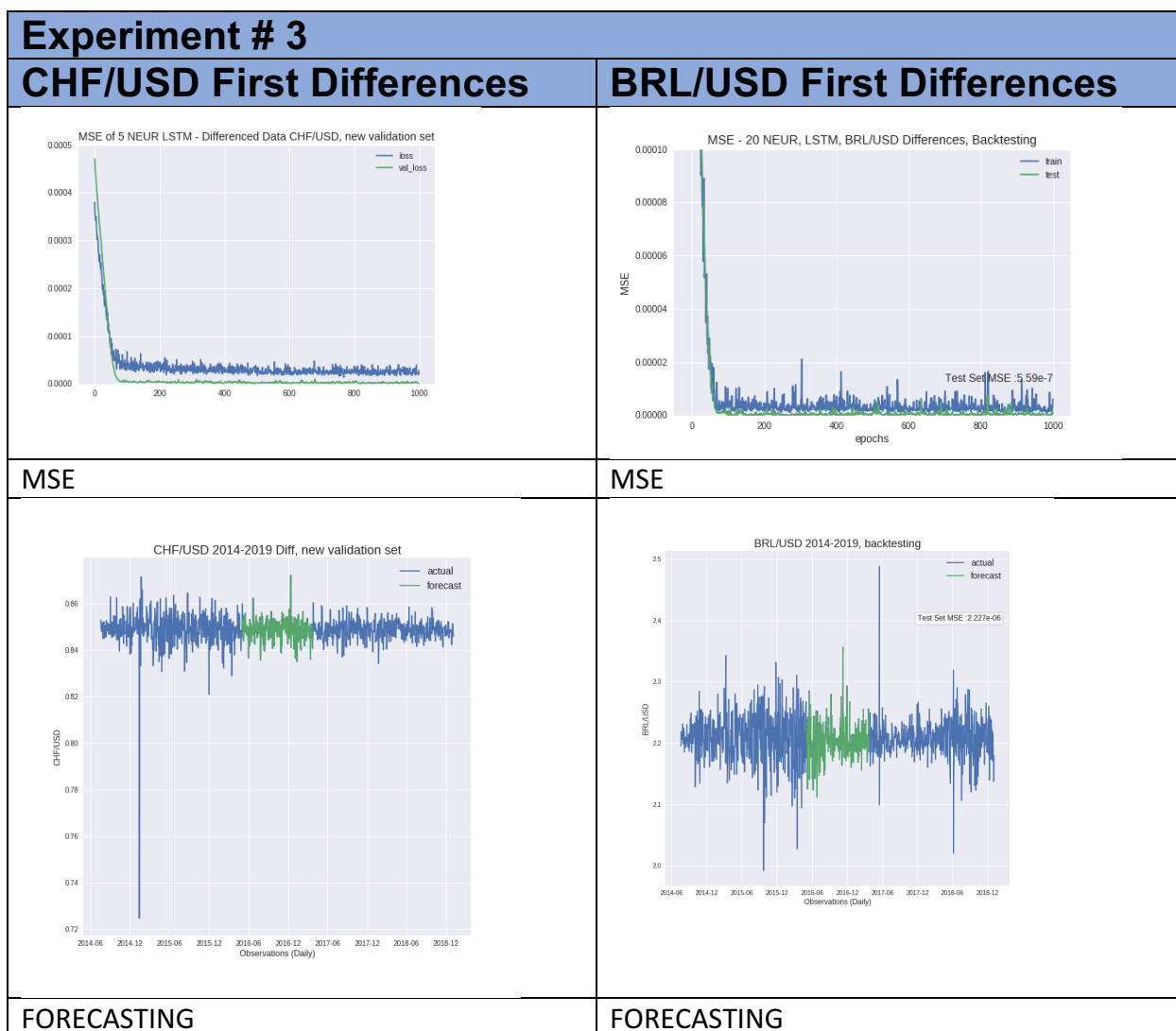
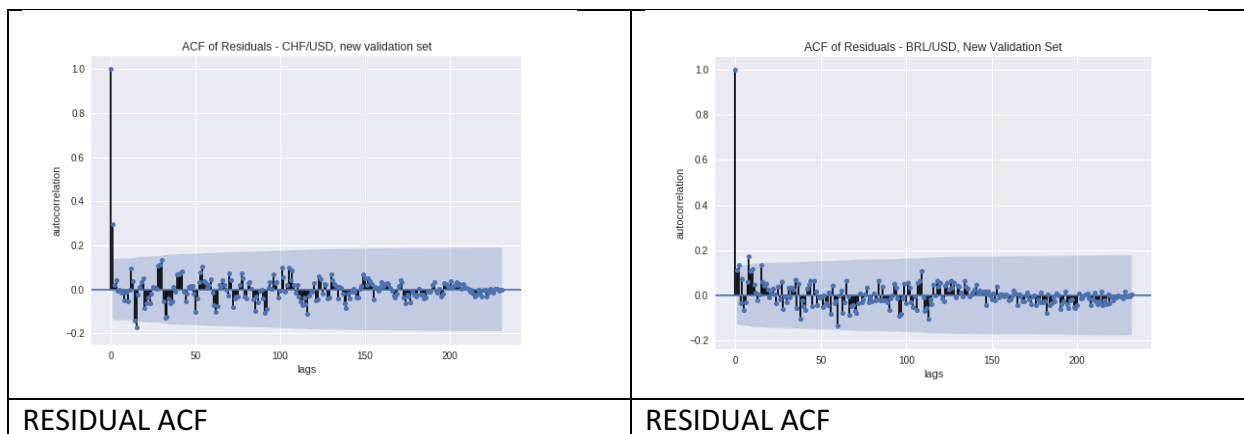
In my second validation approach, I train on the third year of the dataset and back test on second year of data. Here, I also find results consistent with my previous conclusions that the model is producing generalizable, unbiased, and fairly accurate predictions. However, this specification does induce a few wrinkles that are important to note in the analysis of the model performance.

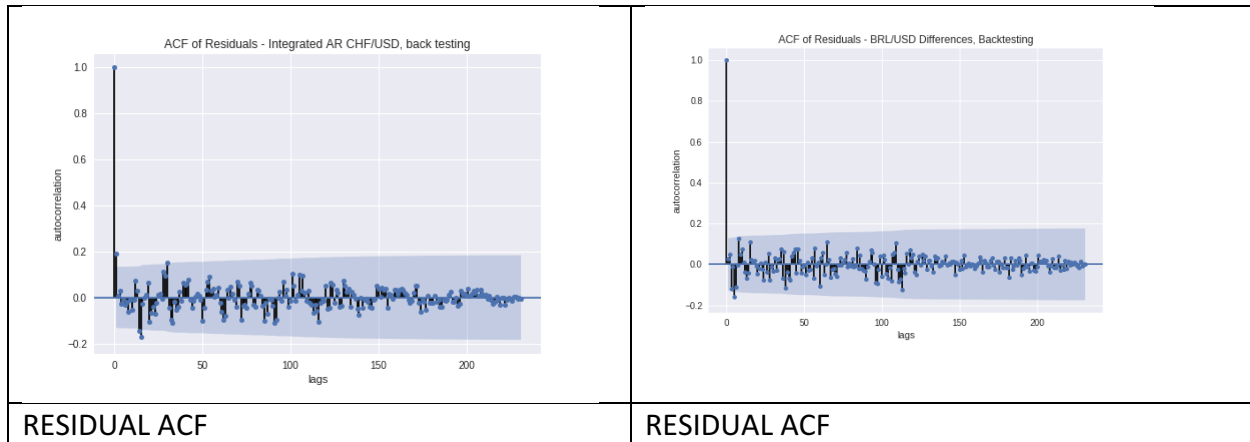
Plots of the model loss decay more slowly and noisily than in other specifications. In the raw time series of CHF/USD and BRL/USD this is likely due to the fact that the unit root in the un-differenced series is causing their means to increase over time. As such, the mean of the

training set data is likely to be different than it is in the test set data. The fact that the mean is likely to change over time is important to consider because our forecasts will be less accurate further into the future as the mean increases or decreases away from what it was during the training period. This suggests that models that train on non-stationary time series would need to be periodically retrained in order to continue to produce accurate forecasts farther into the future.

I do not observe the same discrepancies in the loss plots of the differenced time series and find that they follow the same decay pattern as in the original training and testing period and in the second validation approach. Since the differenced time series are stationary, this supports my assertion that the presence of the unit root can lead to unstable predictions over time if the model is not retrained. Because the differenced series have a constant mean and variance over time, I expect them to perform with the same level of accuracy without periodic retraining.

Experiment # 3	
CHF/USD	BRL/USD
 <p>MSE of 20 NEUR LSTM, new validation set</p> <p>Test Set MSE: 0.000744</p>	 <p>MSE of 20 NEUR LSTM, Backtesting</p> <p>Test Set MSE: 0.000230</p>
MSE	MSE
 <p>CHF/USD 2014-2019, new validation set</p> <p>Test Set MSE: 1.899e-05</p>	 <p>BRL/USD 2014-2019, new validation set</p> <p>Test Set MSE: 0.000106</p>
FORECASTING	FORECASTING





5. Summary and Conclusions

The architecture we used in this study shows that a relatively simple LSTM to Densely connected layer can compute accurate step ahead predictions of the CHF/USD and BRL/USD foreign exchange rates. This was true for the movement in daily foreign exchange rate levels and first differences. The LSTM we employ succeeds in removing serial correlation from the model residuals and makes unbiased time series regression estimates. Furthermore, these results hold in validation on a different subsample and in an explicit back testing approach we developed.

However, our model does lack the complexity to be economically useful. All of the LSTMs trained in this paper produce only a single-step ahead forecasts. Given the wide variety of models that can predict a short time horizon in the future, even the most accurate of forecasts one step ahead are unlikely to extract any additional value over the market. This is due to the fact that market expectations at such a close distance are likely priced into the next day's exchange rate.

Accordingly, extending the architecture tested in this paper to multistep forecasts can increase its economic value for future use. We can also explore two additional tweaks to the architecture that could improve model performance. First, we could stack multiple LSTM layers which could allow the model to uncover more complex temporal relationships in the time series. Since these relationships maybe harder to detect, they may not be already priced into market expectations and could generate additional value for traders. Second, future enhancements to this model should derive an optimal learning rate and test a variety of optimization algorithms. We utilized ADAM in this paper due to its suitability for non-stationary data, but did not provide any tweaks to the learning rate or algorithm beyond what is included in the Keras default settings.

6. Percentage of Code Created

Source	Copied	Modified	Own code
Brownlee LSTM example	4	2	
Jafari NN example	10	5	
Train_Test_IDX			17
SHP_LSTM			12
Neuron Testing For Loop			16
Total	14	7	45

% of derivative code = $14 - 7 / (45 + 14) = 7 / 59 = 12\%$

Note: I count code (whether I wrote or copied) that was pasted multiple times to the same .py file only once. I also do not count functions used from published packages as derivative, but can revise my estimate if needed.

7. References

Brownlee, Jason. "How to Develop LSTM Networks for Time Series Forecasting." *Machine Learning Mastery*, November 14, 2018. <https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>

Brownlee, Jason. "Stacked Long Short-Term Memory Networks." *Machine Learning Mastery*, August 18, 2017. <https://machinelearningmastery.com/stacked-long-short-term-memory-networks/>

Colah. "Understanding LSTM Networks." *Colah.github.com*, August 27, 2015. <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Enders, Walter. *Applied Econometric Time Series*, Fourth Edition. Hoboken: Wiley, 2015.

"Foreign Exchange Rates by Country." *Federal Reserve Economic Database*. July 21, 2019. <https://fred.stlouisfed.org/categories/158>

Hagan et al., *Neural Network Design*. 2nd Edition.

Hochreiter, Sepp and Schmidhuber, Jürgen. "Long Short-Term Memory," *Neural Computation* (1997): 1735-1780.

keras.io/layers/recurrent/

keras.io/getting-started/sequential-model-guide/

Kingma, Diederik and Ba, Jimmy. "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations*, San Diego, 2015, [arXiv:1412.6980](https://arxiv.org/abs/1412.6980)

Segal, Troy. "Forex Market: Who Trades Currency and Why?" *Investopedia.com*. January 23, 2019. <https://www.investopedia.com/articles/forex/11/who-trades-forex-and-why.asp>