Foreign Exchange Rates Using Long Short-Term Memory Networks

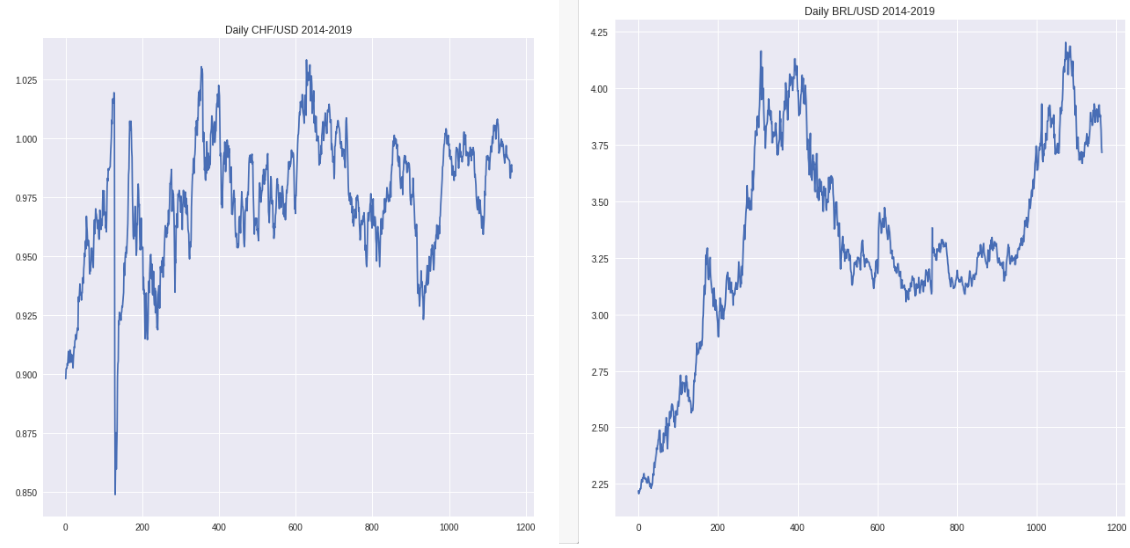
1. INTRODUCTION

This study uses the Long-Short Term Memory recurrent neural network architecture for time series predictions of the Swiss Franc to United States Dollar (CHF/USD) and Brazilian Real to United States Dollar (BRL/USD) foreign exchange rates. Foreign exchange rates represent the value of one currency relative to another in the global market. Accordingly, time series prediction of foreign exchange rates is an economically valuable exercise that presents an opportunity for arbitrage. Expectations of future value drive international trade and investment in a $5.1 Trillion global exchange market.[1] Enhancing the accuracy of these forecasts beyond what is already priced into the market thus has enormous value creating potential in both the short and long term.

2. DESCRIPTION OF DATA

We examine daily foreign exchange rate spot prices of CHF/USD and BRL/USD from July 21, 2014 to January 4, 2019. Both time series are extracted from the Federal Reserve Economic Database (FRED) from July 21, 2014 to January 4, 2019.[2] There are 1,165 observations in each series which equates to roughly 4.65 years-worth of data, given that there are 253 available trading days in a year. Pre-normalization Plots of both time series and summary statistics are shown below.

Figure 1: Plot of CHF/USD (L) and BRL/USD(R) 2014-2019



[1] Troy Segal, "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

[2] "Foreign Exchange Rates By Country," *Federal Reserve Economic Database*, July 21, 2019, https://fred.stlouisfed.org/categories/158

Table 1: Summary Statistics of CHF/USD and BRL/USD

	Mean	Standard Deviation	Min	Max
CHF/USD	.9744	.0269	.8488	1.033
BRL/USD	3.3191	.4437	2.207	3.129

2.1 PREPROCESSING

We linearly interpolate missing values and normalize each time series in order to enhance the training process. Linear interpolation allows us to make the simple assumption that a missing data point falls halfway between the proceeding and succeeding time step. This is a fair assumption, given that the conditional mean at a given time step is considered to be the cumulative sum of the white noise in a random walk data generating process.[3] Normalization is also used to standardize values in the time series so that they fall between zero and one, which will allow the network weights and biases to train more consistently.

2.2 Exploratory Data Analysis

Before constructing the network, it is important to examine the underlying time structure of each foreign exchange rate series. Typically, this requires testing each series for stationarity and the presence of unit roots. A time series is said to be stationary if it has a constant mean and variance across time. Non-stationary implies that either the mean or variance (or both) are time dependent, which will cause the model errors to be serially correlated. This suggests that there is an underlying stochastic process in the data, which will bias estimates and lead to spurious regression.

As is convention, we use the augmented dickey fuller (ADF) test to determine if there is a time dependent association (called a unit root) between an observation and lagged values of itself. The ADF test follows the form:

$$\Delta y_t = \alpha_0 + \gamma_{t-1} \sum_{i=2}^{p} \beta_i \Delta y_{t-i+1} + \varepsilon_t$$

where y_t is the time series variable of interest, $\gamma = -(1 - \sum_{i=1}^{p} \alpha_i)$, $\beta_i = \sum_{j=i}^{p} \alpha_j$, and ε_t is an error term of any number of p lagged values of the time series. The series is said to be weakly stationary if we can reject the null hypothesis that there is a unit root in the dataset.[4]

It is also important to examine the autocorrelation function (ACF) and partial autocorrelation function (PACF) of the time series to determine the effects of past values (autoregressive terms) in predicting the current observation.

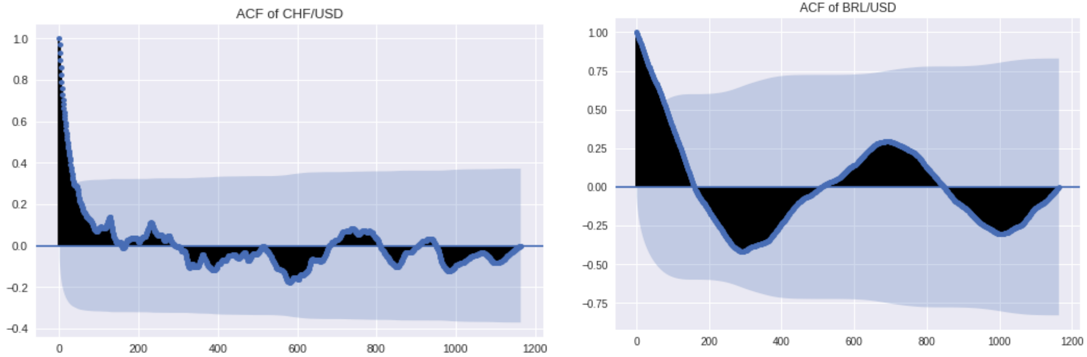
The autocorrelation function represents the correlation between a current observation and past values of itself. A long decay in statistically significant autocorrelations indicates that traditional regression errors are not independent one of another and will lead to bias estimators. This implies non-stationarity.[5]

[3] Walter Enders, *Applied Econometric Time Series*, Fourth Edition. Hoboken: Wiley, 2015, 184.

[4] Ibid., 215-217.

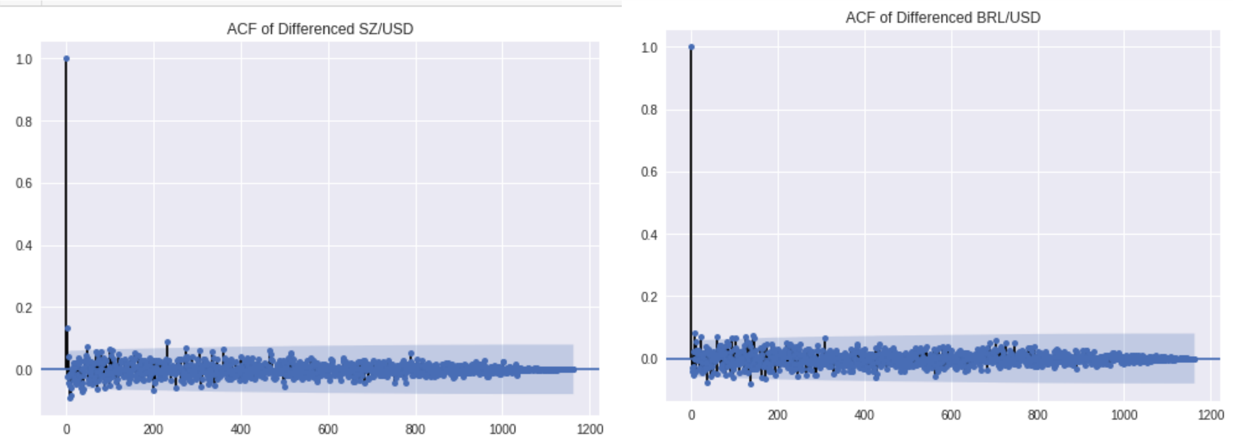
[5] Ibid., 60-63.

Figure 2: Autocorrelation Plots of CHF/USD (L) and BRL/USD (R) Level Changes



ACF plots of both the CHF/USD and BRL/USD rates demonstrate the presence of serial correlation in both time series and the potential for biased estimates. One of way dealing with potential non-stationarity in a dataset is by differencing. Differencing assumes that the change in the variable from observation to observation is approximately white noise with mean zero and constant variance. If a series is stationary in first differences, then we don't observe any serial correlation between current and past observations and can assume that estimates will be unbiased. A plot of the ACF of the first differences of both series appears to show that both CHF/USD and BRL/USD are stationary.

Figure 3: Autocorrelation Plots of CHF/USD (L) and BRL/USD (R) First Differences



Results of ADF tests also confirm that both time series are stationary in first differences. Despite some initial time structure in the ACF plot of the un-differenced data, the ADF tests also state that the CHF/USD series is weakly stationary in the long term.

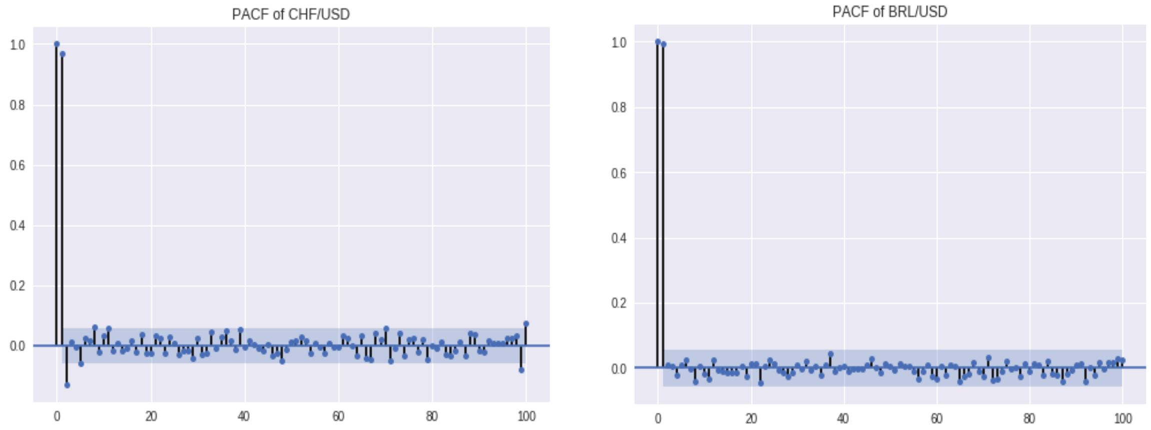
Table 2: ADF Test Statistics in Levels and First Differences.

Time Series	**Levels**	**First Differences**
CHF/USD	-4.1588***	-13.1942***
BRL/USD	-2.3088	-12.5078***

*Note: *, **, *** indicates respective 10%, 5%, and 1% confidence*

Similar to the ACF, the PACF represents the correlation between the time series and lagged values of itself that is not explained by all lower order lags. Statistically significant partial autocorrelations indicate lagged values at a given point in time that are predictive of the current observation. This is known as an autoregressive signature of order p (AR(P)), where p is the last statistically significant lag length in the PACF.[6]

Figure 4: Partial Autocorrelation Plots of CHF/USD (L) and BRL/USD (R)



The sharp decay in in the PACF plots indicates that prior values up to the last statistically significant lag are predictive of future values. From an initial inspection, it appears that CHF/USD is an AR(2) process, while BRL/USD appears to be an AR(1) process.

3. DESCRIPTION OF THE NETWORK

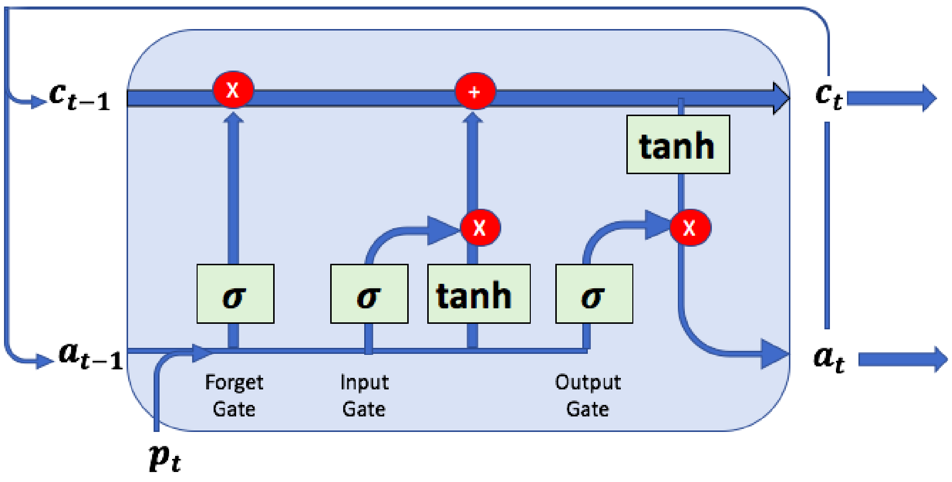
We utilize a Long Short-Term Memory (LSTM) recurrent neural network architecture to develop one step forecasts of the CHF/USD and BRL/USD time series. Originally proposed by Hochreiter et al. in Neural Computation in 1997, LSTM models solve for the "vanishing gradient" problem.[7] In traditional recurrent neural network, back propagation of errors through time can lead to an exponential decay in the network weights. The LSTM recurrent neural network address the vanishing gradient problem by creating a cell of interconnected gate layers to impose a constant error flow through time.[8]

Figure 5: LSTM Network Diagram

[6] Ibid., 64-67.

[7] Sepp Hochreiter and Jurgen Shmid Huber, "Long Short-Term Memory," Neural Computation (1997): 1735-1780.

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



The goal of an LSTM model is to modify the cell state through time C_t and recognize a sequential pattern that is passed through from C(t-1) to C(t). The first gate in the LSTM cell uses a sigmoid transfer function to squash together the current input p(t) with the previous net output a(t-1). This transfer function is known as the forget gate, because the sigmoid output decides what percentage of the previous time information to pass onto the cell state.

$$f_t = \sigma(\boldsymbol{W_f} \cdot [\boldsymbol{a}(\boldsymbol{t}-\mathbf{1}), \boldsymbol{p}(\boldsymbol{t})] + \boldsymbol{b_f}$$

Next, another sigmoid transfer function called the input gate determines which values in the sequence to update, while a hyperbolic tangent is applied to a(t-1) and p(t) to develop new candidate values for the sequence.

$$i_t = \sigma(\boldsymbol{W_i} \cdot [\boldsymbol{a}(\boldsymbol{t}-\mathbf{1}), \boldsymbol{p}(\boldsymbol{t})] + \boldsymbol{b_i}$$
$$\hat{c}_t = \sigma(c \cdot [\boldsymbol{a}(\boldsymbol{t}-\mathbf{1}), \boldsymbol{p}(\boldsymbol{t})] + \boldsymbol{b_c}$$

Then, the network updates the cell state using $f_t, i_t, \hat{c}_t$.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot \hat{c}_t$$

Finally, the output gate applies another sigmoid transfer function to a(t-1) and p(t) to get a signal o_t which determines how much of the hidden state to write to the next hidden state. The dot product between o_t and a hyperbolic tangent of the current cell state creates the output of the LSTM at time t.[9]

$$o_t = \sigma(\boldsymbol{W_o} \cdot [\boldsymbol{a}(\boldsymbol{t}-\mathbf{1}), \boldsymbol{p}(\boldsymbol{t})] + \boldsymbol{b_o}$$
$$a_t = o_t \cdot \tanh(\boldsymbol{C_t})$$

[9] Ibid.

In the context our model, the LSTM receives a three-dimensional array-like input of p(t) of the form (number of samples, time steps, input dimensions).[10] Since we are not specifying a batch size, the first dimension in the array is just the number of observations in the dataset. The number of time steps included in the second dimension is determined by the AR(p) we identified in the PACF function. The dimensionality of the data in the array's third dimension is always one because we are only using one variable, the univariate time series itself, to construct our forecast. The model receives a number of p lagged observations determined by the AR order and the predicted value made at the last time step to make a one-step ahead prediction a(t).

To complete our network architecture, the output of the LSTM cell is connected to a dropout layer to probabilistically reduce the chance of overfitting, and a one neuron densely connected layer with a linear transfer function. The final densely connected layer serves to consolidate the multidimensional LSTM cell output into a single a scalar prediction at each time step.[11]

4. EXPERIMENTAL DESIGN

In this study, we test both a non-integrated and integrated autoregressive prediction for each time series (i.e. four specifications of the LSTM network in total). Traditional autoregressive forecasts require the forecast to be stationary because any correlation between prediction errors over time will bias estimates. Integrated models account for the stationarity by differencing the series to the appropriate order. Hence, an integrated forecast predicts the change in foreign exchange rate differences rather than levels from day to day.

Neural networks are not bound by the same stationarity constraint as a typical autoregressive forecast because the network has the capacity to learn the underlying time structure of the series if it is provided enough data. However, the presence of the time trend will cause the network estimates to eventually become unstable if it is not retrained periodically. This is an important caveat to consider when using an LSTM to make predictions with un-differenced data.

The nature of prediction problems means that we are interested in developing a model that can extrapolate well on out of sample data. As such, we use mean squared error (MSE) as a loss function to evaluate model performance and look at the error of predictions made on the test set compared to the actual values (i.e. the targets). For the same reason, we also use MSE as the cost function to minimize in the training process. After the optimal model configuration has been trained and back tested, we evaluate the autocorrelation function of its residuals. We expect the residuals of an accurate forecast to be uncorrelated through time and be at least weakly stationary.

Because time series data is sequential, we train the model on a given year or years in the dataset and evaluate its performance on a holdout year. For convenience, we leverage a custom function that wrap's scikitlearn's TimeSeries Split function into train and test lists made up of different training and hold out periods. The training and testing periods were modified in three separate experiments with the expectation of obtaining similar model performance. The data for both the Swiss Franc and the Brazilian Real is divided into four years, as mentioned in the data preparation section. The different training and testing periods used for experimentation can be described as follows:

[10] https://keras.io/layers/recurrent/

[11] https://keras.io/getting-started/sequential-model-guide/

Experiment #1
Training begins on each specification of the exchange datasets using the first three years of data and testing on the last year.

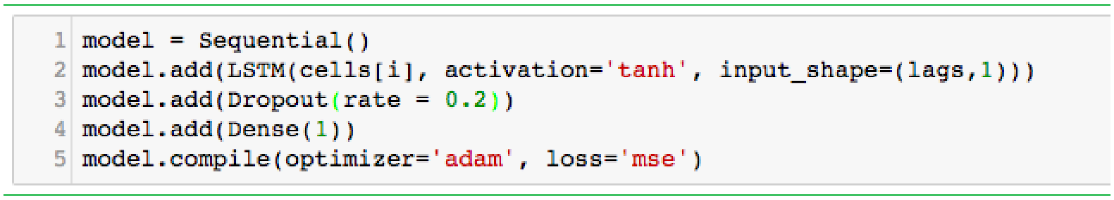
Experiment #2
To check for overfitting, we re-initialize the weights, train on the first two years of data, and test on the third year of the dataset.

Experiment #3
Lastly, the weights are initialized again and the model is back tested using the third year for training and the second for validation.

If forecast mean square errors and plots of training and validation losses appear consistent in each of these three tests, we conclude that the model has not over fit. Results of both validation tests (Test #2 and #3) will be shown in the following section.

To implement the LSTM, we use the Keras library with tesnor flow backend. We order our model sequentially and connect Keras' default LSTM cell to a dropout layer and densely connected layer. We use the adam optimizer and default learning rate in the model due to its efficiency and its popularity among published machine learning research.[12] Figure 6 below shows the Keras layers of the model.

Figure 6: LSTM network in Keras



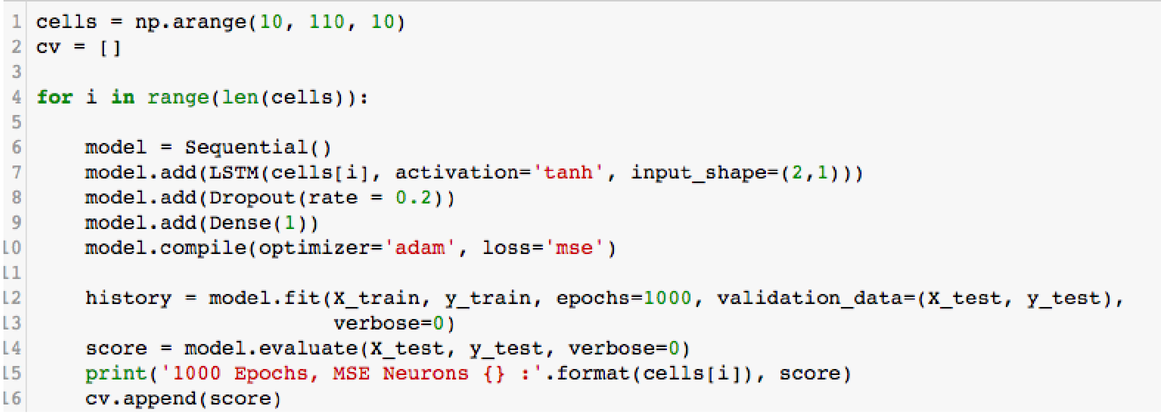
```
model = Sequential()
model.add(LSTM(cells[i], activation='tanh', input_shape=(lags,1)))
model.add(Dropout(rate = 0.2))
model.add(Dense(1))
model.compile(optimizer='adam', loss='mse')
```

We use a for-loop to test configurations of 10 to 100 neurons, increasing by 10, in the LSTM cell to determine the optimal configuration for the LSTM cell. As Hagan et al. suggests, we believe that keeping the number of neurons available to the model below 100 further minimizes the risk of overfitting.[13] Each configuration is trained on 1000 epochs, and the model with the lowest MSE is chosen. In the instance that several neuron configurations produce approximately the same MSE, we selected the one with the lower number of neurons to minimize complexity and risk of overfitting.

Figure 7: LSTM Neuron Selection For-Loop

[12] Diedrick Kingma and Jimmy Ba, "Adam: A Method for Stochastic Optimization," *3rd International Conference for Learning Representations*, San Diego, 2015, **arXiv:1412.6980**

[13] Martin Hagan et al., *Neural Network Design*, 2nd Edition, *27-7*.

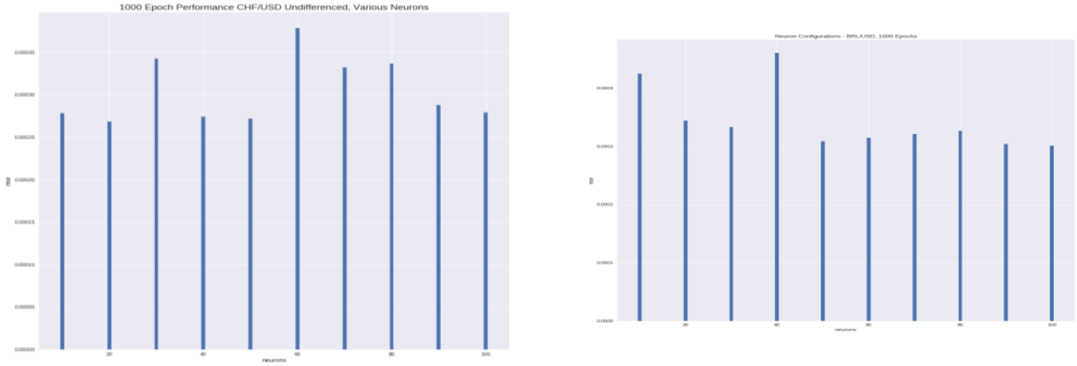


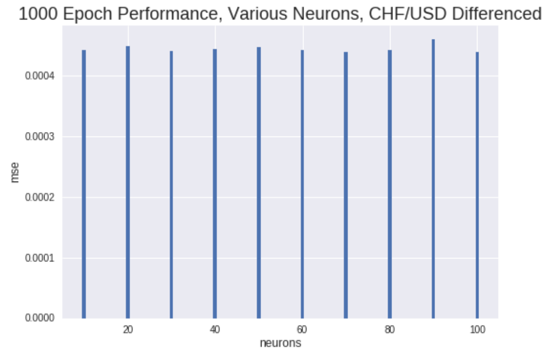
```python
cells = np.arange(10, 110, 10)
cv = []

for i in range(len(cells)):

    model = Sequential()
    model.add(LSTM(cells[i], activation='tanh', input_shape=(2,1)))
    model.add(Dropout(rate = 0.2))
    model.add(Dense(1))
    model.compile(optimizer='adam', loss='mse')

    history = model.fit(X_train, y_train, epochs=1000, validation_data=(X_test, y_test),
                        verbose=0)
    score = model.evaluate(X_test, y_test, verbose=0)
    print('1000 Epochs, MSE Neurons {} :'.format(cells[i]), score)
    cv.append(score)
```

5. RESULTS

We found 20 neurons to produce an optimal forecast estimate for every time series except for the differenced CHF/USD values. Both the CHF/USD and BRL/USD daily observations showed diminishing returns in decreasing MSE when the number of neurons in the LSTM was increased beyond 20. We did not notice any difference between model performance in LSTM configurations with more than 10 neurons and 20 neurons. We 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. We 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 the simpler model would reduce the risk of overfitting.

Figure 7: Optimal Neuron Configurations - *CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right).*





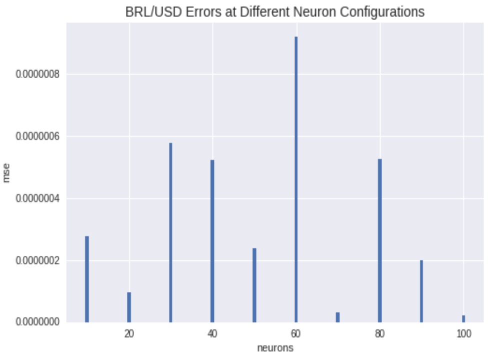


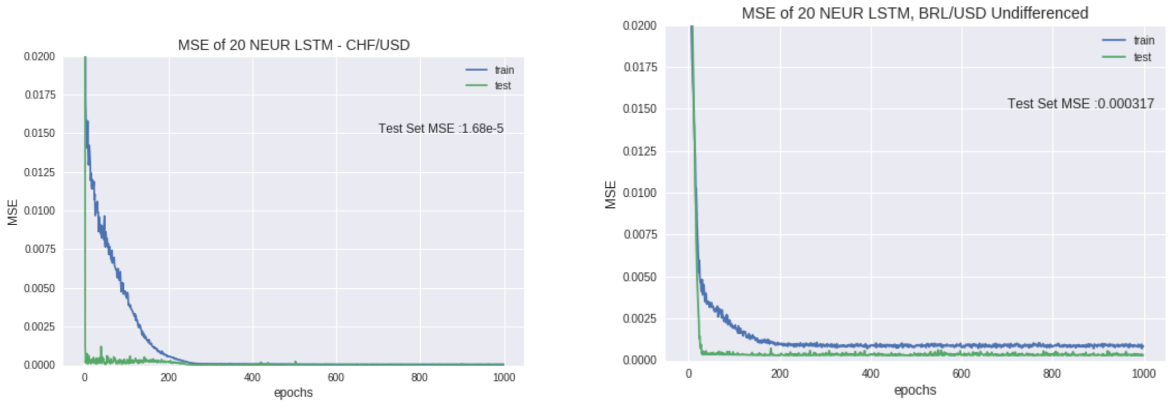
Table 3: Number of Neurons Included in Forecast LSTM

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

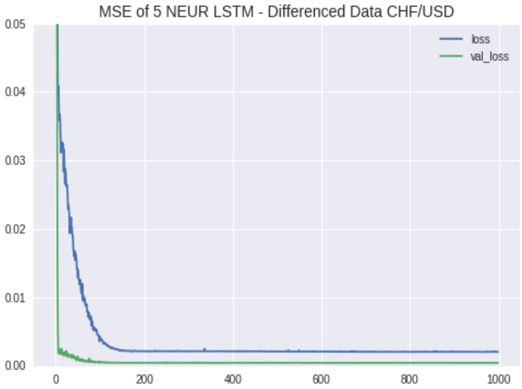
5.1 Experiment # 1 Results

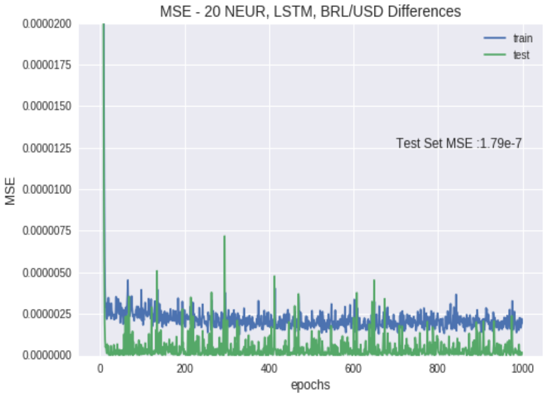
We trained an LSTM on each specification of the foreign exchange rates on the first three years of data and test on the last year of the data using the optimal number of neurons. In each specification, model performance improves on the 3-year training set for the first 200 epochs before stabilizing for the remainder of the training period. Over the same number of epochs, the MSE of the validation set follows a similar decay pattern and does not increase. This is an important distinction, because an increase in the validation error at a certain point in the training process would indicate overfitting and poor generalizability.[14]

***Figure 8: Experiment #1 Model Performance**: 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.*



[14] Amir Jafari, "Lecture 6-1: Generalizability," DATS 6201 *Machine Learning 1* (2019): Slide 13.





We then trained each LSTM specification to generate forecasts on the model test set. From a visual inspection, we observe a fairly close goodness of fit in each specification. This is confirmed by the low MSE in each test set.

Figure 9: Experiment # 1 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.*

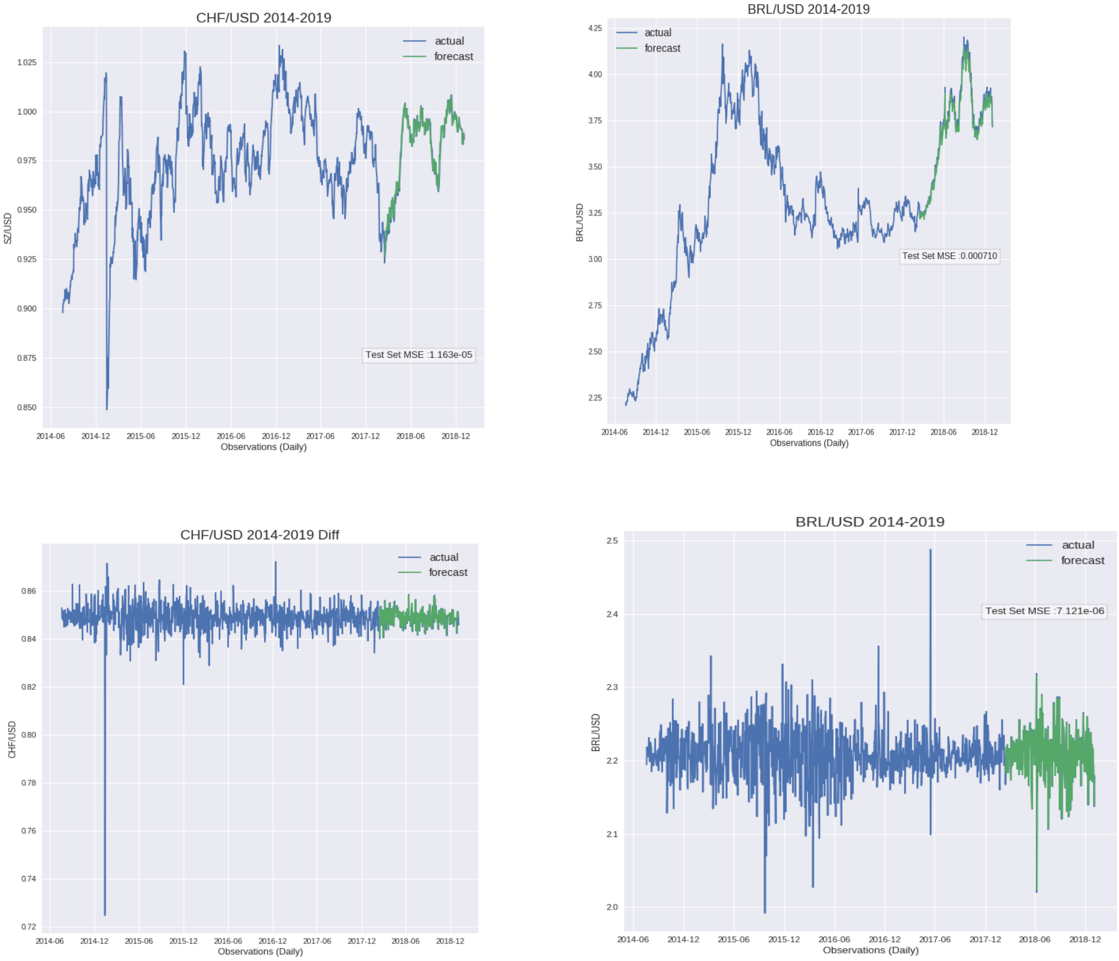


Table 4: *Experiment # 1 Forecasting Mean Squared Errors*

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

Finally, we test for serial correlation in the prediction errors by plotting the ACF of the model residuals (i.e. predicted – actual values).Each ACF plot for the model residuals show a few statistically significant lags after an impulse at zero. However, we find these residuals to be weakly stationary after conducting ADF tests in each time series. This suggests that the forecast estimates made by each model are unbiased.

Figure 10: ACF Plots of Forecast Residuals in Test # 1: *CHF/USD (Top Left), BRL/USD (Top Right), CHF/USD Differences (Bottom Left), BRL/USD Differences (Bottom Right).*

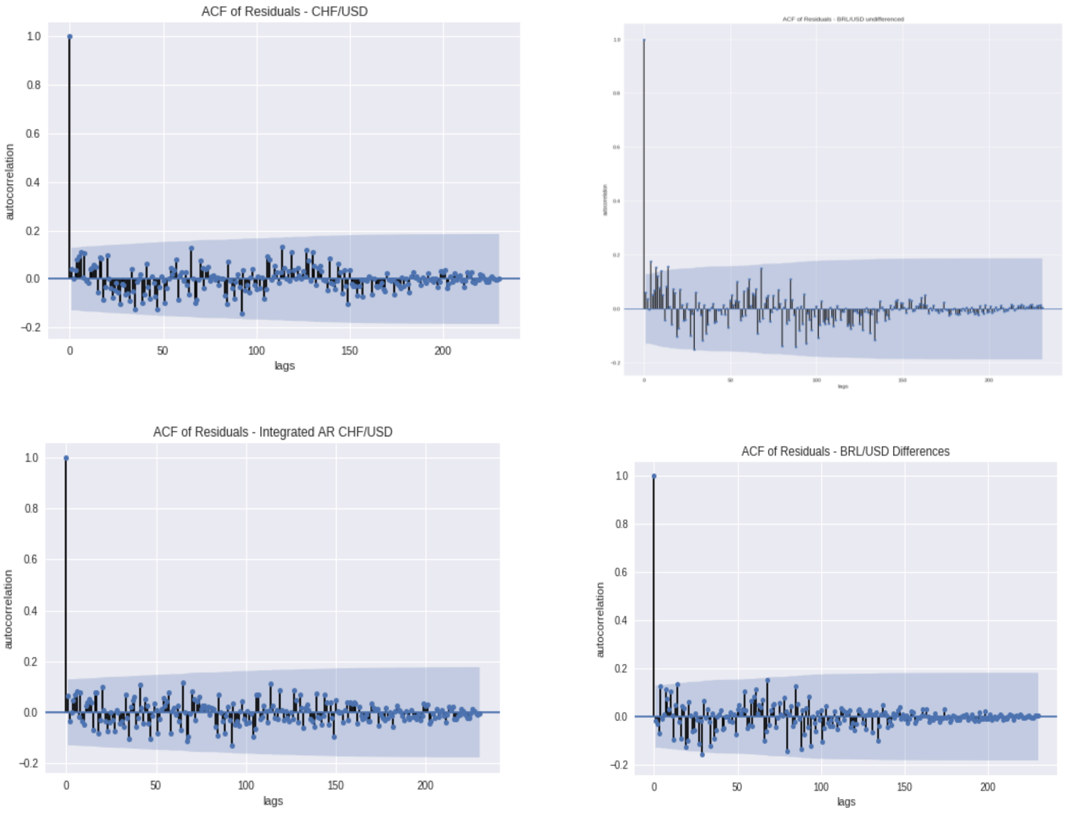


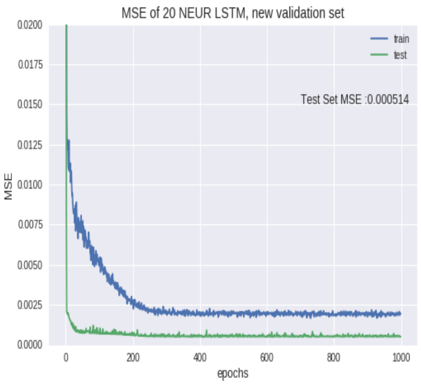
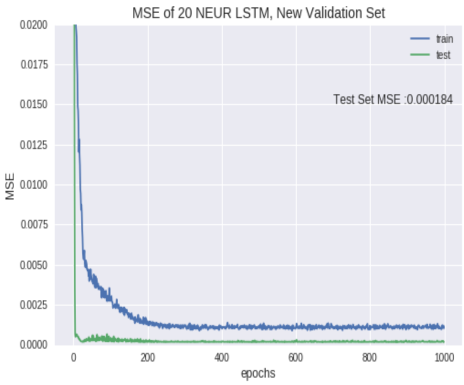
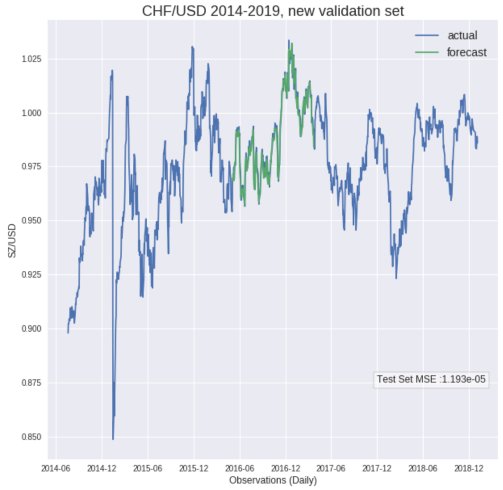
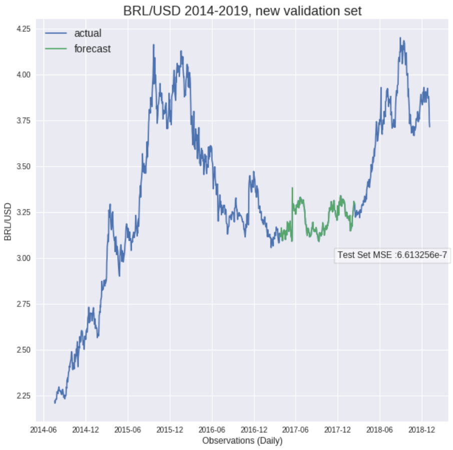
Table 5: *ADF Test Statistics in Levels and First Difference of model residuals.. *, **, *** indicates respective 10%, 5%, and 1% confidence*

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

5.2 Experiment # 2 Results

After the initial training and testing phase, we re-initialize each model and re-evaluate its performance on new subsamples of the dataset. Specifically, in Experiment # 2,we train on the first two years of data and test on the third year of the data set. Each time series we examined produced results consistent with Experiment # 1. Plots of the mean square error function show an exponential decay to zero that stabilizes around 200 epochs, while validation errors also appear to be stable and non-increasing in each specification. This suggests that are model generalizes well when trained and tested on different segments of the dataset. Moreover, the stationary residuals, low forecast MSE, and plots of the predicted values suggest that estimates are both unbiased and accurate.

Figure 11: Experiment # 2 Results

CHF/USD	**BRL/USD**
 	
MSE	MSE
 	
FORECASTING	FORECASTING

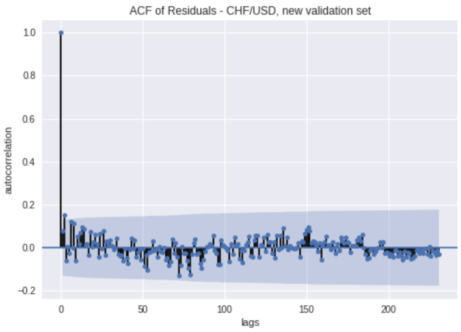
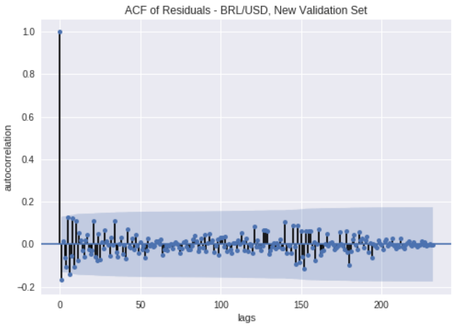
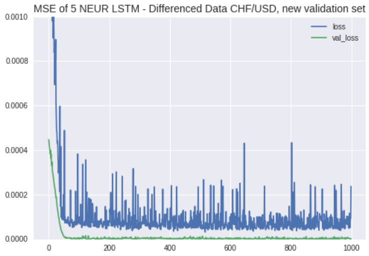
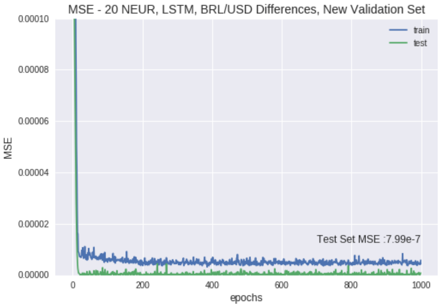
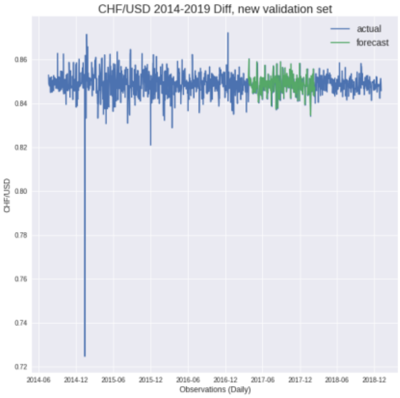
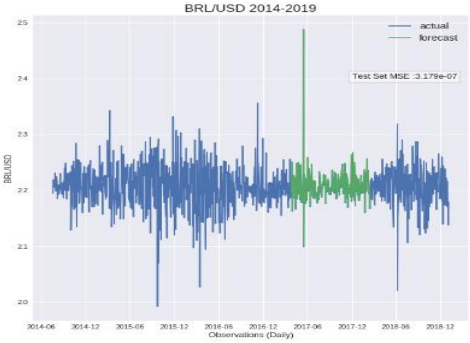
	
RESIDUAL ACF	RESIDUAL ACF

Figure 12: Experiment # 2 Results

CHF/USD First Differences	BRL/USD First Differences
	
MSE	MSE
	
FORECASTING	FORECASTING

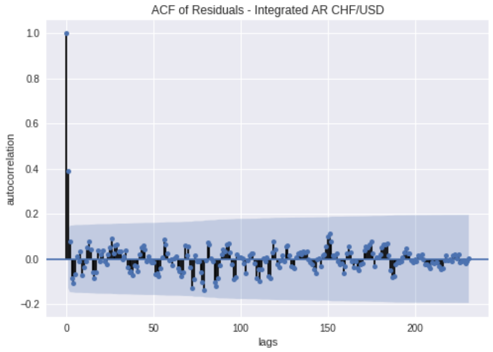
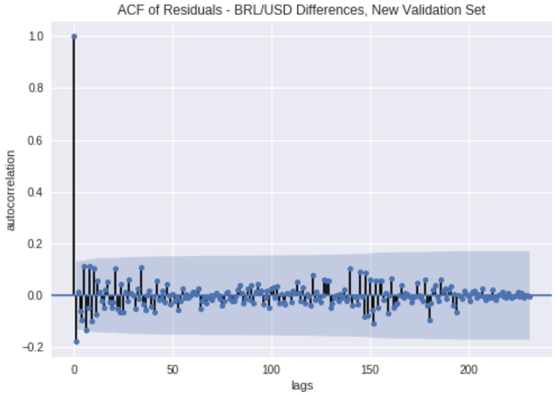
	
Residual ACF	RESIDUAL ACF

Table 6: *ADF Test Statistics in Levels and First Difference of Experiment # 2 model residuals. *, **, *** indicates respective 10%, 5%, and 1% confidence*

Time Series	Levels	First Differences
CHF/USD	-8.13***	-8.64***
BRL/USD	-5.78***	-18.042***

5.3 Experiment # 3 Results

Back testing the data by training on the third year and testing on the second year of the data in Experiment # 3 also produces similar to results to Experiment # 1 and Experiment # 2. However, there are a few items of note in this specification that imply how the models might perform over the long term.

Plots of the model loss decay more slowly and noisily in Experiment # 3 than they do in the first two experiments. For CHF/USD and BRL/USD, this could be because the raw time series are non-stationary and have time dependent mean values. Consequently, the mean of the training set in experiment # 3 is different than it was in the test set, which could bias forecasts in the same direction as the change in means. This implies that the un-differenced time series may become less accurate when making predictions far into the future if the model is not retrained periodically.

The loss plots of the differenced time series follow the same decay pattern that they exhibit in the first two experiments. Because they differenced rates are already stationary, we expect forecasts to perform with the same level of accuracy without requiring the same level of periodic retraining as the raw time series.

Figure 14: Experiment # 3 Results	
CHF/USD	**BRL/USD**

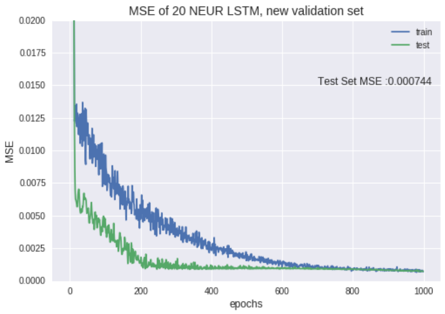
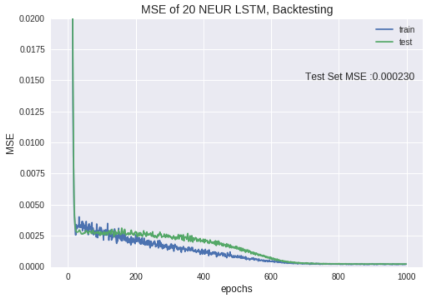
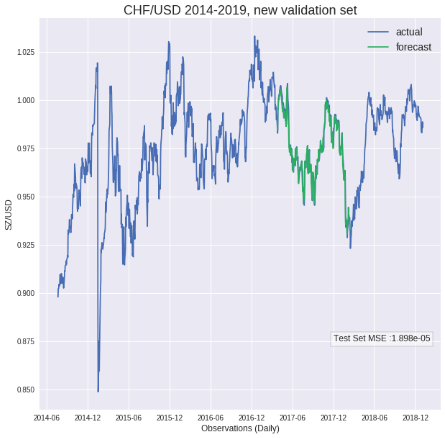
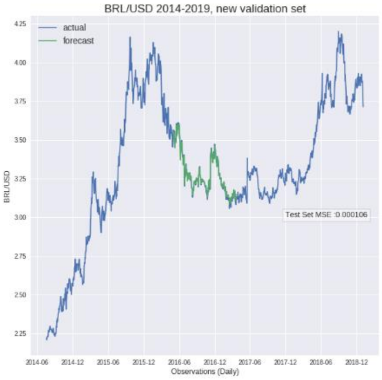
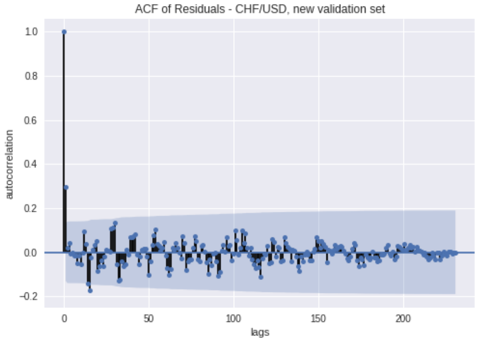
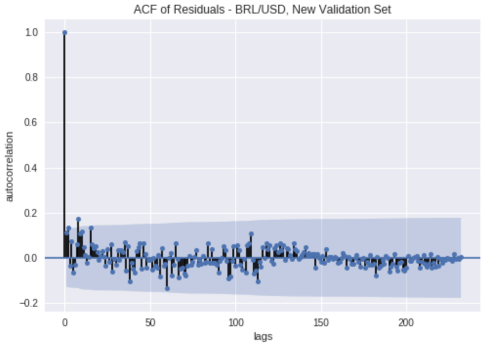
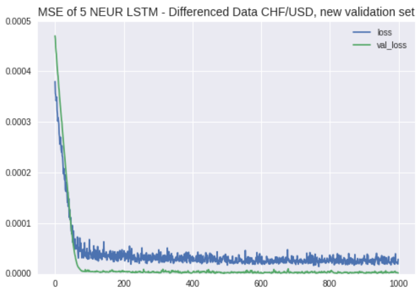
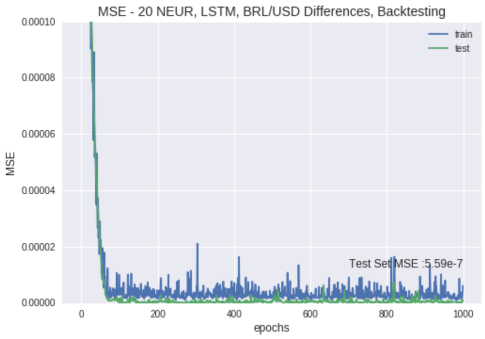
	
MSE	MSE
	
FORECASTING	FORECASTING
	
RESIDUAL ACF	RESIDUAL ACF

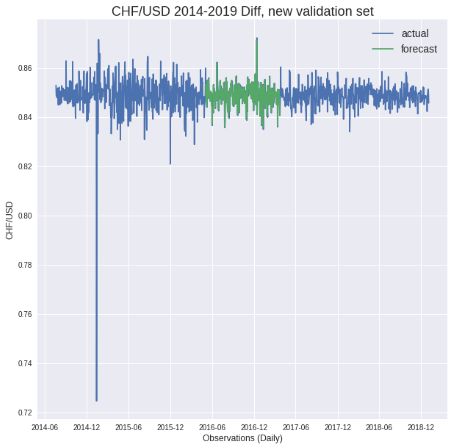
Figure 15: Experiment # 3	
CHF/USD First Differences	**BRL/USD First Differences**



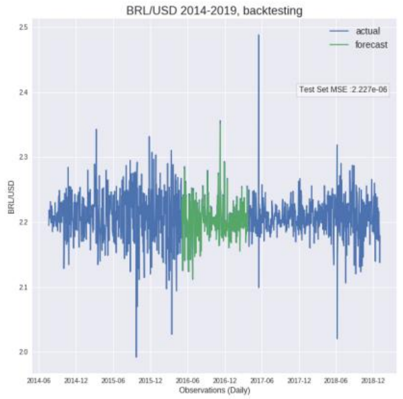
MSE



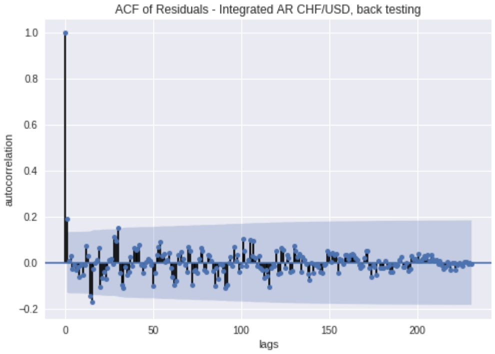
MSE



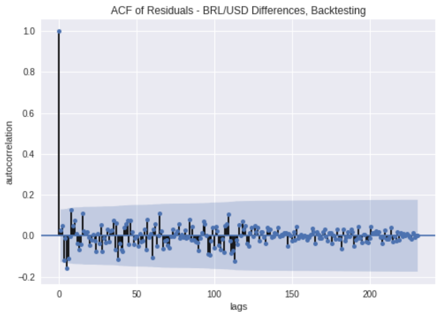
FORECASTING



FORECASTING



RESIDUAL ACF



RESIDUAL ACF

***Table 7:** ADF Test Statistics in Levels and First Difference of Experiment # 3 model residuals. *, **, *** indicates respective 10%, 5%, and 1% confidence*

Time Series	Levels	First Differences
CHF/USD	-11.156***	-12.37***
BRL/USD	-8.86***	-8.13***

6. SUMMARY AND CONCLUSIONS

The simple LSTM architecture used in this study shows that the model was a fairly accurate predictor of the step-ahead foreign exchange rates of the CHF/USD and BRL/USD daily levels and first differences. Moreover, stationary residuals, consistent loss plots, and fairly low forecast MSE suggests that the model estimates are both unbiased and accurate. This is supported by the consistency of the results when training on a different subsample of the data (Experiment # 2) and in back testing (Experiment # 3).

Despite these promising results, the forecasts produced in this study have limited economic value. The single step ahead predictions we created show the value of a foreign exchange rate in levels or differences only one day into the future. Since these predictions are somewhat easy to predict with a variety of models including a random walk process, one day movements are likely to be priced into the foreign exchange rate by the market. Though we show the accuracy and potential of LSTMs for time series prediction, we do not provide evidence of its economic value in making predictions.

Our model could be improved and made more valuable for economic prediction of exchange rates in a few distinct ways. First, the model architecture could be adjusted to create multi-step forecasts at a further point into the future. This could enhance our confidence in its ability to extrapolate and make long term predictions. Second, we did not test for the optimal depth of the LSTM network in this paper. Other studies have shown that adding additional LSTM layers can increase prediction accuracy while reducing the risk of over fit.[15] Finally, we did not test for an optimal learning rate or compare optimization algorithms in this study, which should be incorporated into future implementations of the model to improve the training process.

7. References

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 Shmid Huber, Jurgen. “Long Short-Term Memory,” Neural Computation (1997): 1735-1780.

[15] Jason Brownlee, “Stacked Long Short-Term Memory Networks,” *Machine Learning Mastery,* August 18, 2017. https://machinelearningmastery.com/stacked-long-short-term-memory-networks/

keras.io/layers/recurrent/

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

Kingma, Diedrick and Ba, Jimmy. “Adam: A Method for Stochastic Optimization,” *3rd International Conference for Learning Representations,* San Diego, 2015, **arXiv: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

8. Appendix – Computer Listings

- Python 3.6.2
- Numpy (For Linear Algebra operations)
- Pandas (DataFrame, Series)
- Scikit Learn (MinMaxScaler, TimeSeriesSplit)
- Statsmodels (plot_acf, plot_pacf, adfuller)
- Matplotlib (for plotting)
- Seaborn (for plotting on top of matplotlib)
- Keras (for Machine Learning training)
- Tensorflow (Keras backend)