Vinit Patel

Time Series Prediction using RNN Architectures Report

In this project I experimented with different architectures of Recurrent Neural Networks (RNNs) for Time Series Prediction. The data I used consist of Daily EURUSD prices. The architectures, I will be implementing and comparing are:

1. LSTM with Residual Connections
2. Attention Based LSTM

The code: https://github.com/vinit97/Attention-Based-RNN-for-Time-Series-Prediction

1st Step - Data Prep:

Preprocessing

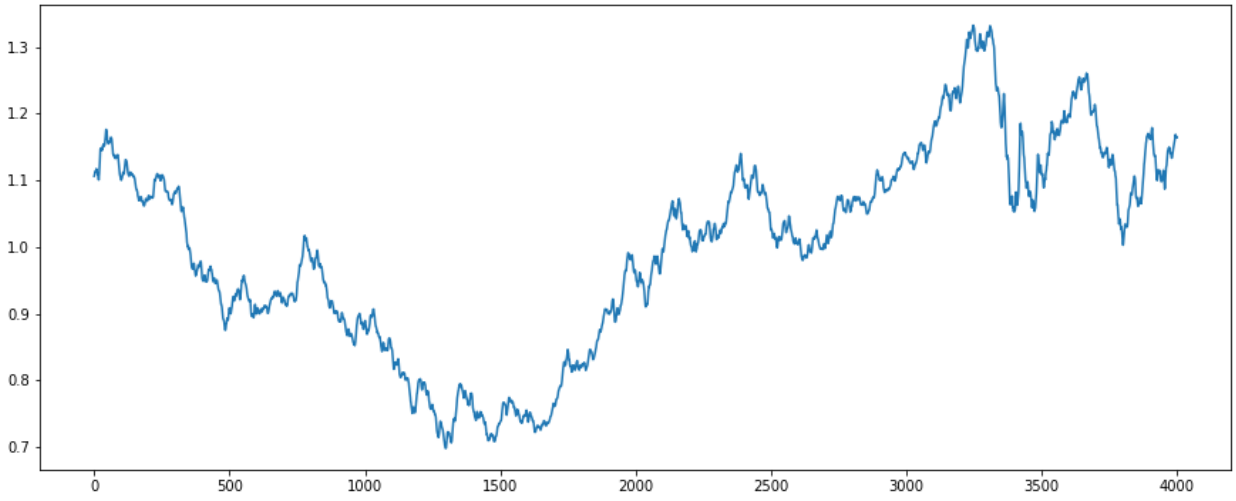
Preprocessing the data is extremely important when using neural nets.

In 2018, Uber Engineering won the M4 forecasting competition using a hybrid model. The hybrid merged an Exponential Smoothing Forecasting Method and Neural Networks, namely, Holt-Winters and Recurrent Neural Networks. Holt-Winters formula works quite well with seasonality-based time series data, as it included seasonality multiplicative coefficients [1]. However, in the forex market the seasonality trends are rare. Therefore, I decided to preprocess the data using an Exponential Moving Average, which can be derived from the Single Exponential Smoothing Formula algorithm.

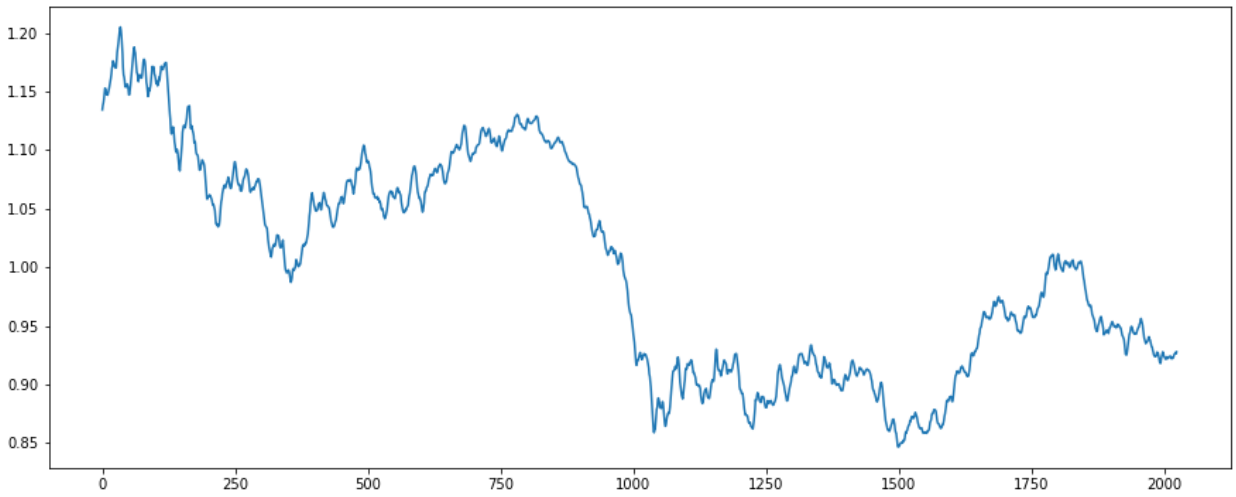
$$S_t = \alpha \mathrm{p_t} + (1-\alpha)\mathrm{S_{t-1}}\,, \quad 0 < \alpha < 1$$

$$\mathrm{S_t} = \alpha(\mathrm{p_t} + (1-\alpha)p_{t-1} + (1-\alpha)^2 \mathrm{p_{t-2}} + \cdots)$$

where α is the smoothing factor, p_t is the price and S_t are exponentially smoothed price at time t. I choose to be α to be 0.2. Then the test and train data were separately normalized by their respective means.

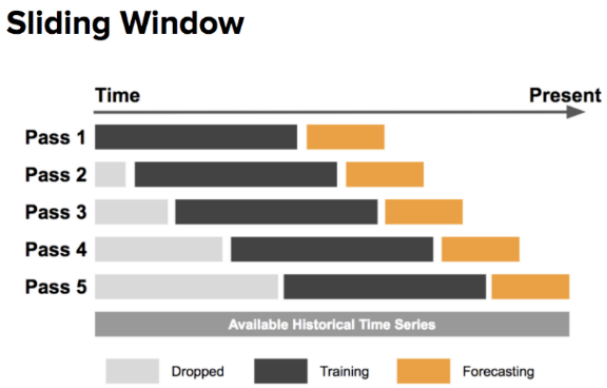


Preprocessed Train Data: 1995-08-23 to 2011-03-15



Preprocessed Test Data: 2011-03-16 to 2018-12-31

Time Series Models must be trained in a specific way. I will be using the sliding model for traning both the models. For example, I used a fixed traning window of size **5** and Forecasting window of size **1** for the LSTM with residual connection model. This means given **5** days of past prices, the model could forecast the price movement for the next day. The sliding window method can be seen in the image below.



[1] Uber Engineering: Forecasting Methods

Evaluation Metrics:

Mean Absolute Error (MAE):

$$MAE = \sum_{t=1}^{n} \frac{|y_t - p_t|}{n}$$

Mean Absolute Percentage Error (MAPE):

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|y_t - p_t|}{y_t}$$

, where y_t is the actual value, p_t is the prediction at time t

LSTM with Residual Connections

LSTM is a recurrent neural network which was first proposed in 1997. The LSTM unit solves the gradient exploding/vanishing problem that early RNN units had. The yellow bars represent the residual connections, aka skip connection which carries information across 1 layer (in this model).

Model:

$h_t^1 = f(h_{t-1}^1, x_t)$
$h_t^2 = f(h_{t-1}^2, h_t^1) + h_t^1$
$h_t^3 = f(h_{t-1}^2, h_t^2)$
$y = f_1(h_t^3)$

f is a LSTM unit. The general form of a LSTM unit is:

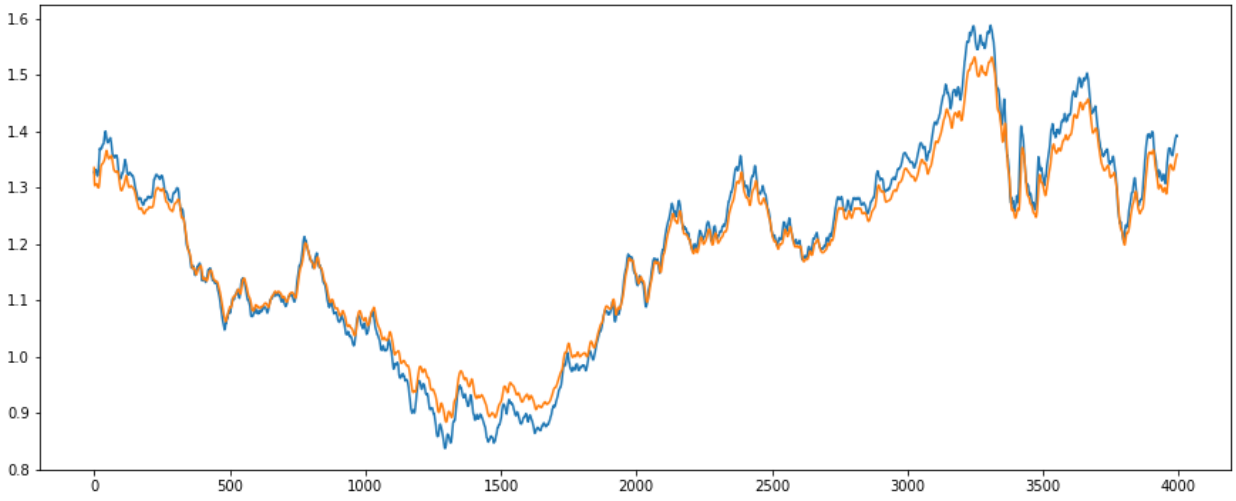
$$f_t = \sigma\big(W_f[h_{t-1}; x_t] + b_f\big)$$
$$i_t = \sigma(W_i[h_{t-1}; x_t] + b_i)$$
$$o_t = \sigma(W_o[h_{t-1}; x_t] + b_o)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s[h_{t-1}; x_t] + b_s)$$
$$h_t = o_t \odot tanh(s_t)$$

$x_t \in R^T, y \in R^f$
$h_t, b_f, b_i, b_o, b_s \in R^m$ is the hidden state at time t and all the biases
$\mathrm{W_f}, \mathrm{W_i}, \mathrm{W_o}, \mathrm{W_s} \in \mathrm{R}^{m \times m}$ are the weight matrices
$\sigma, \odot$ are the sigmoid function and element-wise multiplication respectively

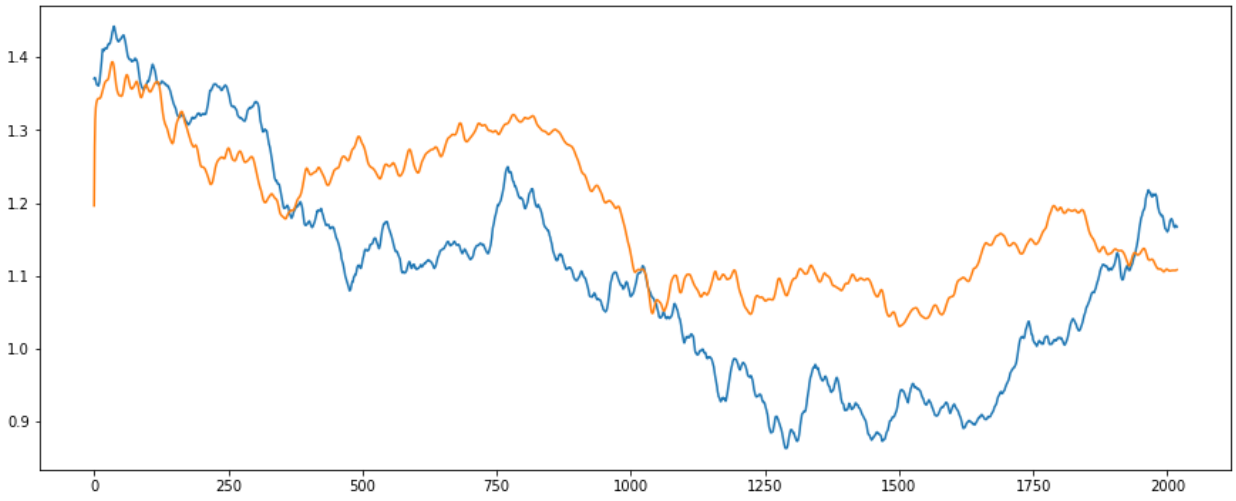
Optimized model parameters:

LSTM Units m = 128, Traning Window T = 5, Forecast Window f = 1, and Loss Function = MSE.

Used entire train dataset from 1995-08-23 to 2011-03-15 to train the model.



Blue: Real Train Data, Orange: Train Prediction



Blue: Real Test Data, Orange: Test Prediction

The model fits the train data well. But on the unseen data, the model cannot consistently predict the future prices.

MAE: 0.0803
MAPE: 8.5362 %

Attention Based LSTM

Over the past few years, the attention mechanism has been used extensively in major machine learning breakthroughs. The idea is quite simple. Instead to focusing on the entire time series, we only pay attention to certain parts of the data and then make a prediction. The model I implemented was partially based on the paper Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction [2]. In this model, we only apply attention to the inputs.

The model follows:

Given input x, hidden state h_{t-1}, cell state s_{t-1},

$$score(\mathrm{h_{t-1}}, \mathrm{s_{t-1}}) = \mathrm{v_a^T}\, tanh(\mathrm{W_a}[\mathrm{h_{t-1}}; \mathrm{s_{t-1}}] + W_x x + b_a)$$

$$\alpha_\mathrm{t} = \frac{\exp\big(\mathrm{score}(\mathrm{h_{t-1}}, \mathrm{s_{t-1}})\big)}{\sum_{\mathrm{i=1}}^{\mathrm{t}} \exp(\mathrm{score}(\mathrm{h_{i-1}}, \mathrm{s_{i-1}})\}}$$

$$\bar{x} = (\alpha_1 x_1, \alpha_2 x_2, \dots, \alpha_t x_t)$$

Then we pass the attended inputs to the LSTM.

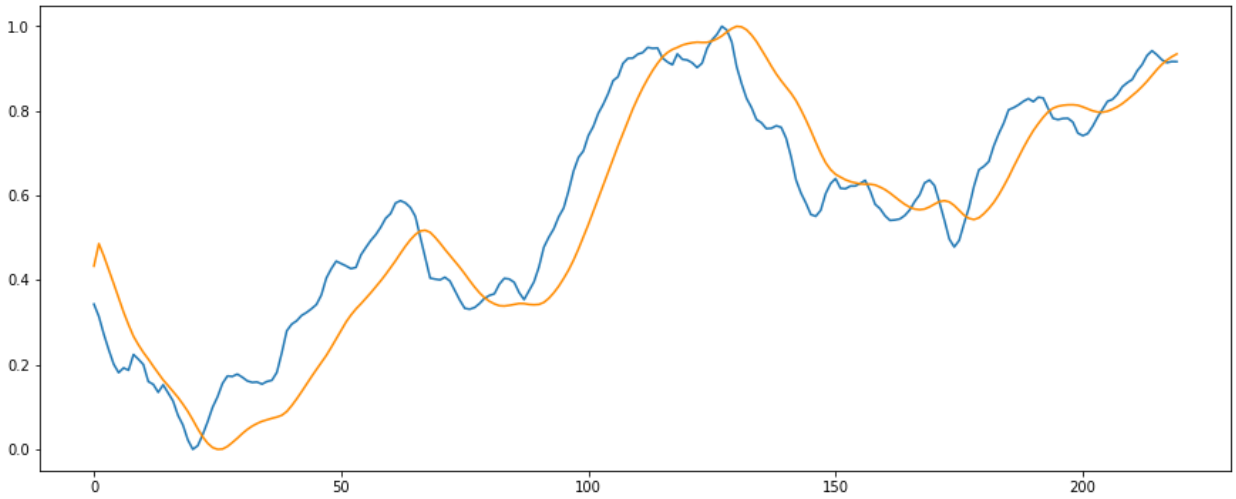
$$f_t = \sigma\big(W_f[h_{t-1}; \bar{x_t}] + b_f\big)$$
$$i_t = \sigma(W_i[h_{t-1}; \bar{x_t}] + b_i)$$
$$o_t = \sigma(W_o[h_{t-1}; \bar{x_t}] + b_o)$$
$$s_t = f_t \odot s_{t-1} + i_t \odot \tanh(W_s[h_{t-1}; \bar{x_t}] + b_s)$$
$$h_t = o_t \odot tanh(s_t)$$

$x_t \in R^T, y \in R^f$
$v_a \in R^T, W_a \in R^{T*m}, W_x \in R^{T\times T}, b_a \in R^T$, the attention weights and biases
$h_t, b_f, b_i, b_o, b_s \in R^m$ is the hidden state at time t and all the biases
$\mathrm{W_f}, \mathrm{W_i}, \mathrm{W_o}, \mathrm{W_s} \in \mathrm{R}^{m\times m}$ are the weight matrices

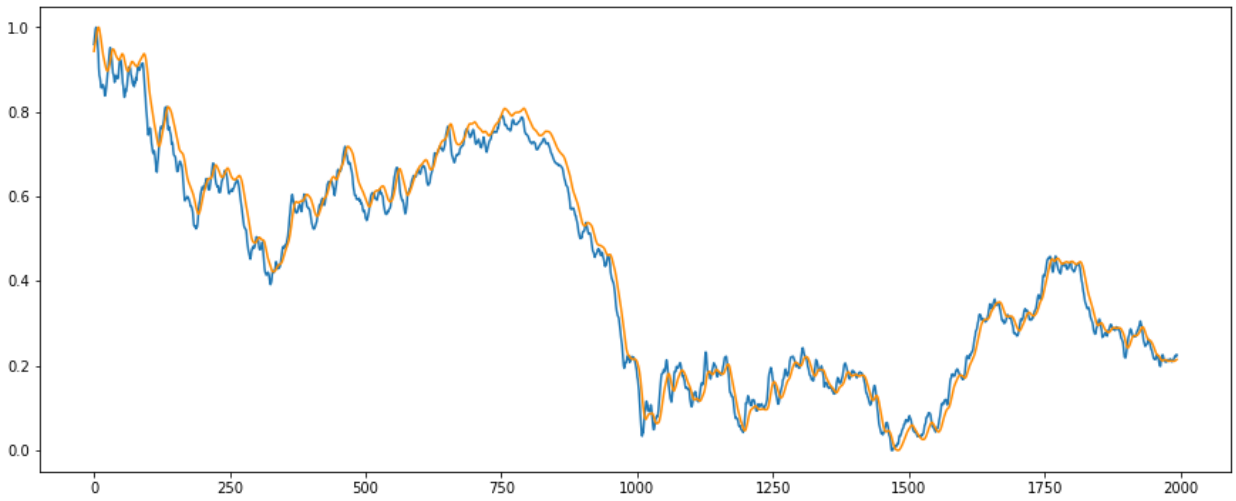
Optimized model parameters:

LSTM Units m = 128, Traning Window T = 30, Forecast Window f = 1, and Loss Function = MSE.

Used train dataset from 2010-03-30 to 2011-03-15 to train the model. That's 1 years compared to LSTM 16 years of data.

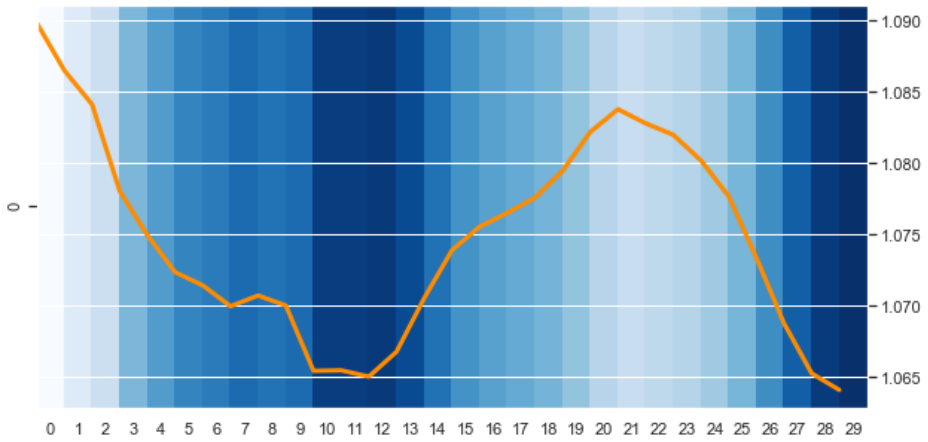


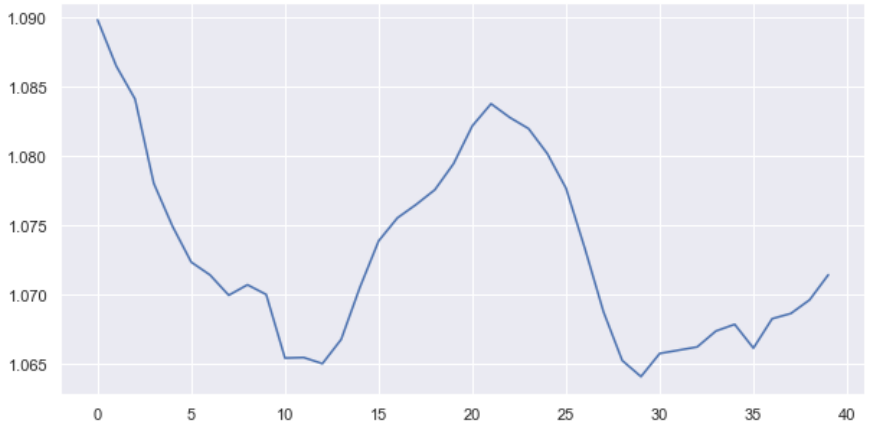
Blue: Real Train Data, Orange: Train Prediction



Blue: Real Test Data, Orange: Test Prediction

The results are quite surprising. The model was accuracy able to predict next days price for the next 8 years, given only 1 years' worth of data. To understand, what the model is doing, we need to look at its attention weights given some data.





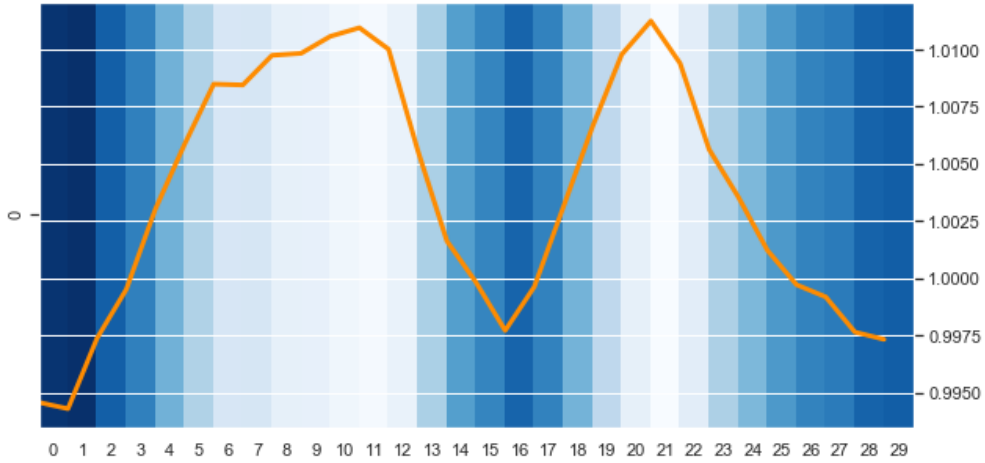
Given random selected sections of the test data, we will look at different ways the model accurately predicts the data.

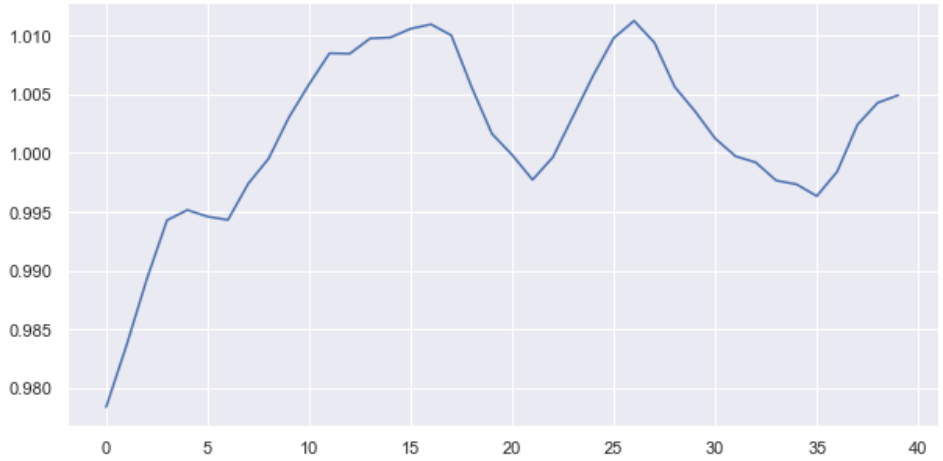
Dark Blue means the model is paying **high attention** to the section. **Light Blue** indicates that model thinks that section is not relevant (**low attention**).

In the top figure to the left, the model pays high attention to areas [26-30] and [8-14]. The price went up after hitting a low at 12, indicating the market has support at price 1.065.

Therefore, model acknowledges the support level, and predicts the price should go up from 1.065 because it went up last time. That's exactly what happens after t = 30, the price starts crawling upwards.

Similar story for the 2 images below.

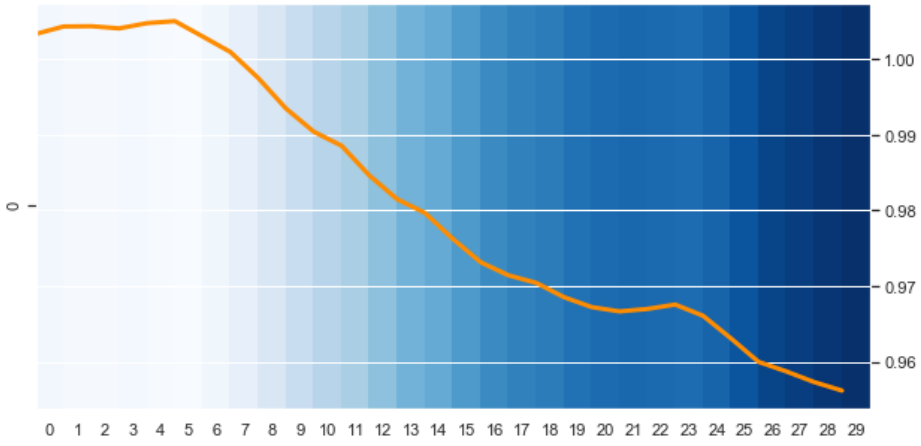


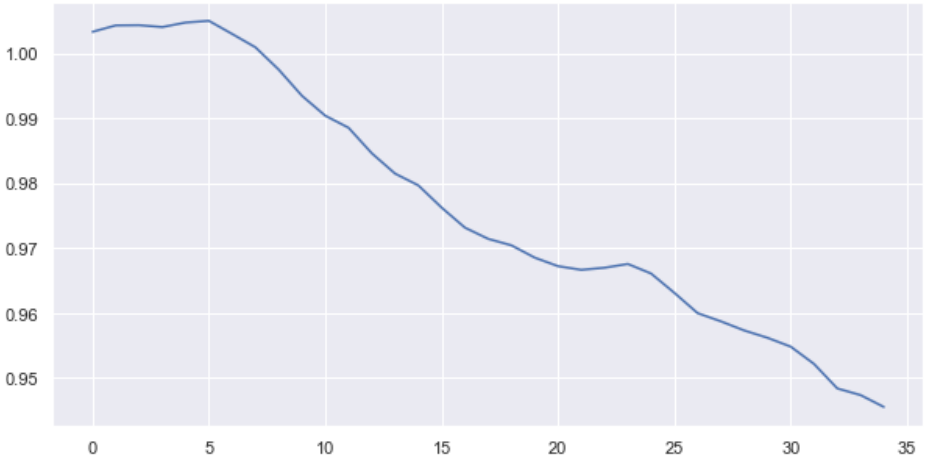


The model believe that the support should hold at 0.9975 and the price should go up, therefore its paying attention to the places ([0-3], [14-17], [27,28]), where the price turns long.

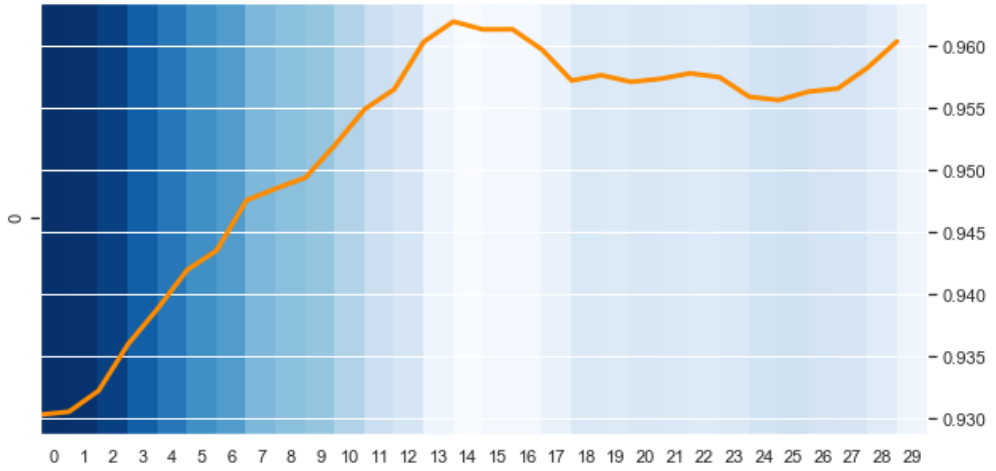
The 2 images to right, shows an example of trend continuation.

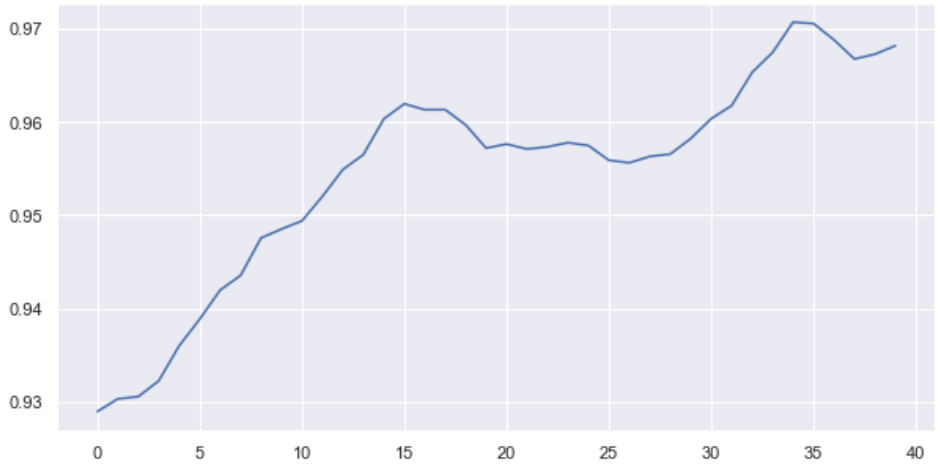
The price is short and has gone down from 1 to 0.96. Therefore, the model is paying higher attention to the latest price series, indicating future downward price movement.





This last example below shows model predicting the breaking of resistance. There was a resistance at 0.960, preventing it from going upward (t = 14). This is particularly tricky, since the price after going to 0.960 at t = 12 went down to a low of 0.955. However, this dip was small. Therefore, the model accurately predicts the price going long and breaking the 0.960 resistance barrier. As we the see from the bottom image, that's exactly what happens.





MAE: 0.02462221

MAPE: 1.73296783 %

MAPE is significantly lower for the attention model than LSTM with residual networks. However, if we increase the forecast size for both models, both MAE and MAPE increase, due to high variance. I believe it's since the forex market is extremely volatile, even with exponential smoothing.

Resources:

[1] M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model. https://eng.uber.com/m4-forecasting-competition/

[2] Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., & Cottrell, G.W. (2017). A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. *IJCAI*.

[3] Attention? Attention! https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html