Deep Learning for Currency Pair Prediction

Dmitri Koltsov[1], Amirsadra Mohseni[2], Linxuan Liu[3]
University of California, Riverside
{[1]dkolt002, [2]amohs002, [3]lliu163}@ucr.edu

1 Introduction

Background

A currency pair is the quotation of two different currencies, with the value of one currency being quoted against the other. In this report, we utilize six major currency pairs including EUR/USD, GBP/USD, USD/JPY, EUR/GBP, EUR/JPY, GBP/JPY as our datasets, focusing on the currency pair data retrieved in 1-minute intervals. The output of our models is whether a curious currency trader should buy, sell, or hold at this particular moment in time. We would like to remind our readers that this project is only meant to serve as a proof of concept and is not meant to be deployed in a real-life scenario involving valuable assets.

Our models use the data from the year 2017 to predict the year 2018, the year 2018 to predict 2019, and 2019 to predict 2020 using three machine learning methods optimized for time series prediction including a vanilla Recurrent Neural Network (RNN), Long-Short-Term-Memory (LSTM) [1], and Gated Recurrent Units (GRU). The code we used to produce these results is freely available, in the Python language, here: [**GITHUB LINK**] All models are implemented using the PyTorch [2] library.

Related Work

Galeshchuk et.al [3], give a brief overview of prediction problems and some methods to solve that such as econometric models, time series models, ANN, and DNN. Nguyen [4] mentions several regression models can be useful in deep learning and contribute to our project. Vukovic [5] uses natural network models for prediction on USD/EUR currency pairs. He also provides some advanced ideas about data collection and deep learning models that yield good results in prediction.

2 Materials and Methods

Data Acquisition

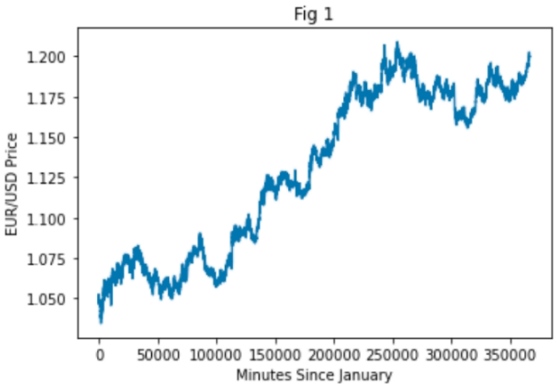
In this report, we use the Forex history currency pair data retrieved at 1-minute intervals obtained from HistData [6] as our dataset. We hypothesized that using only the year 2020 to predict the currency trends will prove too sensitive and the results may not be significant enough. Therefore, we train our models on the data from 2017 to predict 2018, then 2018 to predict 2019 and, finally, 2019 to predict 2020. Each data file contains the timestamp and ratio for a given currency pair. We combined data matching timestamps. Ultimately, we have 4 datasets for each year 2017, 2018, 2019, and 2020. Each dataset has the following structure:

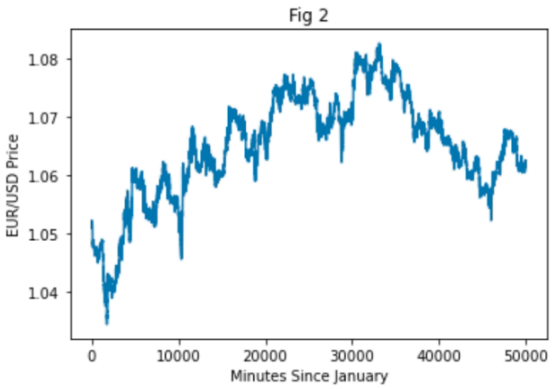
- Columns represent all six currency pairs: EUR/USD, GBP/USD, USD/JPY, EUR/GBP, EUR/JPY, GBP/JPY

- Each row represents the currency pairs' ratio for a given timestamp separated by every 1 minute

Preliminaries

Fig 1 Shows the currency pair EUR/USD over the year 2017 and fig 2 shows the price of EUR versus USD from January through the middle of February of the same year.





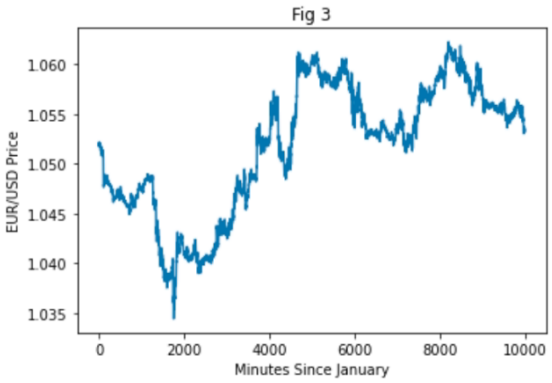


Fig 3 shows the same price pair for the first 10 days of 2017. Observe that there are smaller trends within the long-term trends and that even though a long-term trend may increase, smaller trends may fluctuate within this main trend. We assume that long-term trends (months) depend on “out of market” situations such as economy, politics, natural disasters, etc. In contrast, short-term trends depend on “inside the market” situations. When traders buy particular stocks more, the price rises, and when they sell those stocks, the price falls. This assumption is not far from the truth, though oversimplified for this project. Our predicting strategy is based on this simplifying assumption. We analyze market behavior and try to predict short-term trends. In the stock market, such a strategy is called “shaving” or “scalping.” This is not what professional brokers use, however, it is employed extensively by beginner individual traders.

Predicting Strategy

Two of the required hyper-parameters to train our models are the period for which we are going to predict a trend, called “prediction length,” and our target difference of price, called “price jump.” For each timestamp, we look in the future for the duration of “prediction length” and check if the price goes up or down for the “price jump” within this period. We create a target variable “y” for our datasets based upon these observations. If the price goes below the threshold of “price jump,” we predict “sell.” If the price increases, we predict “buy.” If the price stays within the “price jump” limits for “prediction length,” we predict “hold.”

Accordingly, our one-hot-encoded target variable “y” is illustrated as follows:

“sell”: [1, 0, 0] “hold”: [0, 1, 0] “buy”: [0, 0, 1]

To evaluate the model, we could not simply compare “ŷ,” our prediction, and “y,” the true event because we remain indifferent if the model correctly predicts “hold” since this prediction does not yield any profit or loss. The solution then is to compare “ŷ” and “y” and count “wrong” if we predict “buy” when we have to “sell” or vice versa. Conversely, we count “right” if we predict “buy” or “sell” correctly.

Example of using the model:

1. Define 2 hyper-parameters experimentally: "price jump" and "prediction length." These are different depending on the currency pairs

2. At any timestamp, the model predicts what to do. For example, the model says "buy." We buy a fixed amount of units of EUR/USD pairs
3. Now we anticipate one of the 3 following events:
    a. Price goes up for "price jump"
    b. Price goes down for "price jump"
    c. Neither of the previous two happen during "prediction length"
4. Whichever happens first – we sell.

If we guess right, we profit by a **fixed price**, if we guess wrong, we lose the same **fixed price**. If the price does not go up or down, we would sustain, approximately, no financial gain or loss[1].

3 Experiments

For training and evaluating we use the following hyper-parameters:

- Sequence length – How deep back we are going to perform back-propagation
- Prediction length – The period for which we are going to predict trends
- Price jump – The price difference between "buy" and "sell" points
- Hidden size – The dimension of the hidden layer
- Number of hidden layers
- Learning rate
- Epochs
- Batch size

After creating the "y" variable as mentioned before, we may start the learning and evaluation processes.

RNN

First we designed an RNN (many to many) with following parameters:

- Sequence length = 1000
- Prediction length = 10000
- Price jump = 0.01
- Hidden size = 42
- Number of layers = 3
- Learning rate = 1e-4
- Epochs = 2
- Batch size = 1

Initially, we choose a small price jump just to check how the model works. We trained the model on the datasets for three years: 2017, 2018, and 2019 and tested it on 2020. Results:

- Right: 112749
- Wrong: 75682

We decided to use a different approach. We trained the model on 2017 and tested on 2018 and performed the same routine to train the model on the dataset for one year and test on the next. The results were much worse. "Right" and "wrong" were approximately even. Sometimes "right" was higher, and sometimes "wrong" was higher.

[1] It is true that whenever we buy or sell, we may have to pay some fees. There are also margins etc. but as we mentioned before, we would like to disregard these costs and instead explore theoretical possibilities of our machine learning models.

So we decided to change some of the hyperparameters. We increased "sequence length", "prediction length" and "price jump" by a factor of 2:

- sequence length = 2000
- prediction length = 20000
- price jump = 0.02

We hypothesized that making "price jump" more prominent will enable our model to discriminate more confidently and increasing "sequence length" will make the model more sensitive to past events. These new parameters improved our results:

Training on: Testing on:	2017 2018	2018 2019	2019 2020
Right	78409	19248	82129
Wrong	50739	15141	61335

We considered these experiments sufficient and moved on to the next model using the same hyperparameters.

LSTM

LSTM is an improvement of RNN. Briefly, LSTM deals with past events better than an RNN, avoiding the problem of vanishing gradient descent. Hence, to improve our prediction and explore other architectures as an educational endeavor, we opted to model an LSTM. We used the same hyper-parameters we used for the final well-performing RNN. The "rights" and "wrongs" were approximately equal and in some cases not in our favor:

- Right: 71828
- Wrong: 75003

We concluded that LSTM takes past events into account better than RNN and perhaps that a "sequence length" of 2000 was too high for RNN because of the vanishing gradient problem (we used sigmoid as non-linearity). It may have been the case that the RNN could not "remember" so far back, and truly used a smaller "sequence length" than we specified. We decided to reduce the "sequence length" of our LSTM to 512 and the results became slightly better:

Training on: Testing on:	2017 2018	2018 2019	2019 2020
Right	86002	17973	79703
Wrong	75113	12146	66151

These results led us to believe that we may be on the right track.

After experimenting with different "sequence lengths" we concluded that a "sequence length" of 256 or 512 gives us approximately equal and best results among other "sequence length" choices.

GRU

GRU is much similar in functionality as an LSTM, only that it uses 2 gates instead of 3 and therefore is computationally more efficient and trains faster. However, whether it is better or worse than LSTM is still a subject of debate.

We used different "sequence length" = 128, 256, 512 and 1024. We get the best results with "sequence length" = 256:

Training on: Testing on:	2017 2018	2018 2019	2019 2020
Right	98438	28777	106069
Wrong	76710	14488	80696

4 Conclusions

We used 2 types of RNN: "many to many" and "many to one." "Many to many" produced better results. We presume this is because we used randomly sampled sequences (since fully training "many to one" takes much more time). In other words, we trained "many to one" on much fewer samples than "many to many."

We used RNN many to many, RNN many to one, LSTM, and GRU, and between these, GRU and RNN produced better and more stable results. We can use our model to decide whether to buy or sell EUR/USD currency pairs.

Testing

There is a possibility that our model is coincidentally right for the given data. As we mentioned, we used different hyper-parameters in different cases and picked them experimentally.
For testing purposes, we picked the hyper-parameters which gives us better performance in general among all models, and used them on different currency pairs without any changes or tuning. We used EUR/GBP and GBP/USD and you can see the results below:

EUR/GBP

	RNN many-to-many	RNN many-to-one	LSTM	GRU
17-18	Right 24637 Wrong 13863	Right 21143 Wrong 14907	Right 15740 Wrong 18992	Right 27372 Wrong 14093
18-19	Right 61299 Wrong 48633	Right 52642 Wrong 43277	Right 58515 Wrong 52813	Right 63851 Wrong 54709
19-20	Right 52509 Wrong 51153	Right 54451 Wrong 47379	Right 54451 Wrong 47379	Right 66360 Wrong 52358

GBP/USD

	RNN many-to-many	RNN many-to-one	LSTM	GRU
17-18	Right 167648 Wrong 136252	Right 72185 Wrong 91795	Right 86890 Wrong 98574	Right 166273 Wrong 133532
18-19	Right 138612 Wrong 111615	Right 138612 Wrong 90078	Right 106753 Wrong 90303	Right 123373 Wrong 108532
19-20	Right 173447 Wrong 130161	Right 117130 Wrong 87774	Right 118130 Wrong 86774	Right 144063 Wrong 117598

As we can see, the results are less stable for all but RNN (many to many) and GRU.

Models were tuned for different currency pairs and each currency pair has its characteristics. The situation is not like 1 dollar = 1 euro = 1 pound and so on. This is why "price jump" should be chosen individually. Furthermore, different currencies may have different dynamics that affect their prices according to their countries" economic structures. Some economies may react faster, some may have "amortization" mechanisms so their currencies react slower. Thus, we may need to choose different "sequence lengths" and "prediction lengths" for different pairs.

Ultimately, our test results show that RNN, LSTM, and GRU models that are not exhaustively tuned for particular currency pair can still provide positive results. This supports our assumption that short-term trends can be predicted based on price history.
Instructions of using the model would be:

1. Gover more years of data.
2. Choose a currency pair.
3. Tune hyper-parameters for this currency pair uning training and validation sets.
4. Test the model on a test set.
5. Use the model in real time to predict this currency pair behavior on the market (make some profit).

Again, in this project we are not creating optimized model for use in the market, we are exploring possibility for recurrent neural networks predict trends based on price history.

Supplementary Experiments

Stock market behaves differently right after opening and right before closing (weekends, holidays). We modified our data and included feature indicated periods before opening and closing.
We also experimented with different prediction lengths.
Prediction length = 1024

Training on: Testing on:	2017 2018	2018 2019	2019 2020
Right	12	0	186
Wrong	0	0	12

With a short prediction period the model became much more cautious, predicting hold more often, but also more accurate.

Prediction length = 4096

Training on: Testing on:	2017 2018	2018 2019	2019 2020
Right	13886	487	16779
Wrong	7409	56	9942

Naturally by increasing prediction length more chances for price change it's value by 'price jump'.

We noticed that during the training process loss stopped reducing at some point. So we tried to stop training when loss stops reducing. Results became much worse. We concluded that with more data despite training loss stays approximately the same model still keeps training and adjusting and validation accuracy improves.

5 References

[1] "Long short-term memory." *Wikipedia*, en.wikipedia.org/wiki/Long_short-term_memory. Accessed 30 May 2021.

[2] *PyTorch*. pytorch.org

[3] Galeshchuk, Svitlana, and Sumitra Mukherjee. "Deep learning for predictions in emerging currency markets." *International Conference on Agents and Artificial Intelligence*. Vol. 2. SCITEPRESS, 2017.

[4] Nguyen, Andrew. "Exchange Rate Prediction: Machine Learning with 5 Regression Models." *towards data science*, Medium, 20 May 2020, towardsdatascience.com/exchange-rate-prediction-machine-learning-with-5-regression-models-d7a3192531d. Accessed 27 Apr. 2021.

[5] Vyklyuk, Yaroslav, Darko Vukovic, and Ana Jovanovic. "Forex prediction with neural network: USD/EUR currency pair." *Актуальні проблеми економіки* 10 (2013): 261-273.

[6] *HistData*. https://www.histdata.com/. Accessed 27 Apr. 2021.