# ECE247: USD-EUR Exchange Rate

Kevin Chen
504863921

Paokuan Chin
504059777

Stathis Megas
104683465

## Abstract

*Several experiments completed in this work support the view that NN algorithms outperform traditional machine learning algorithms in economic series forecasting. Moreover, modern DL architectures, like GRU and LSTM, seem to outperform older architectures like CNN, especially when the former are combined (via ensemble methods or preprocessing) with traditional machine learning algorithms.*

## 1. Introduction

Time series analysis is the primary forecasting tool of economists due to its versatility and significant impact. From early on, machine learning techniques have been successfully applied to forecast time series, and by now there exists a vast literature. The recent advent of Deep Learning (DL) has found its way into economics research and has contributed models that significantly outperform their traditional machine learning counterparts [2, 3]. With the aim of understanding the application of DL methods in economics and finance, we pursue a comparative study of different neural network (NN) architectures in forecasting the USD-EUR exchange rate.

## 2. Experiments and Results

In this study, we constructed and trained NNs to predict future values of the USD-EUR exchange rate, given historical data leading up to that day. Since our data is ordered temporally, a very natural architecture is the recurrent neural network (RNN). We also considered convolutional neural networks (CNNs) since the filters may be able to pick out temporally-organized features in the time series.

The working hypothesis for all our experiments was that the RNN can outperform non-NN techniques. In order to gauge the success of our NNs, we picked a simple linear regression model to use as a benchmark: given the exchange rate over the past $k$ days, take the least-squares fit and extrapolate for future days. $k$ is a hyperparameter of this model that is optimized on the training set.

### 2.1. Data

Our data was obtained from [1]. The primary time series utilized in this study is the USD-EUR exchange rate, i.e. the daily closing price of the Euro in US Dollars, from Jan. 1, 1970 to Dec. 31, 2020. We also obtained additional time series as motivated by economic theory, including other exchange rates (e.g. USD-Pound), government bond yields, consumer price indices, stock indices, commodity prices, and unemployment rates. In total, we obtained 54 unique time series. Certain time series were reported monthly or yearly, so these were upsampled by repeating previous values. We normalized all time series into the interval $[0, 1]$.

An observation we made early in our study was that the USD-EUR exchange rate did not have uniform statistics over time. Practically, this meant that a model training on the years 1970–2015 and validating on the years 2016–2019 would often find a validation loss lower than the training loss. This apparent pathology can be explained by the fact that the years 2016–2019 are simply more "predictable" than other years. By drawing testing and validation sets randomly across the entire time series, this pathology disappears. Since the NN architectures we explored trained on subintervals of one large time series, we had to ensure that the training set was mutually exclusive to these two sets.

### 2.2. Next-day forecast

As a proof of concept, we started by training a simple RNN to forecast the next-day USD-EUR exchange rate. The RNN takes in a time window of 30 days and predicts the exchange rate on the 31st day. The model was optimized on the mean squared error (MSE) loss between the predicted and actual price.

| *Model* | *Average MSE on Testing* |
|---|---|
| Benchmark | $3.04 \times 10^{-5}$ |
| LSTM | $2.83 \times 10^{-5}$ |
| GRU | $2.40 \times 10^{-5}$ |

Both LSTM and GRU architectures were able to beat the benchmark model. The LSTM model performed worse and took almost twice as long to train, so we primarily used GRU layers in future models.

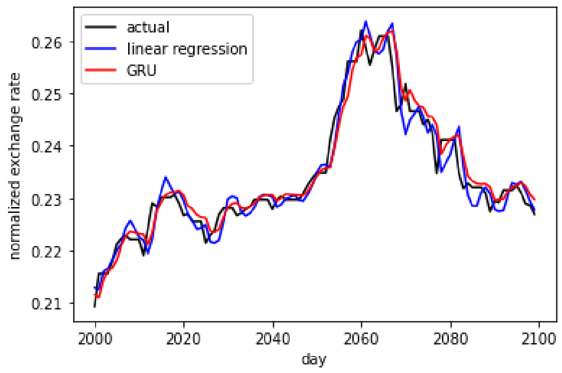

Figure 1. Next-day forecasts.

One could argue that beating linear regression at next-day forecasts is not an impressive feat, as the MSE errors are so small relative to the actual exchange rate. Moreover, there are modifications to linear regression that can improve performance, such as the exponential moving average. However, models based on average values fail at predicting longer trends in the exchange rate. It was thus our goal to see how well we could make long-term predictions, hopefully taking the other time series into consideration.

### 2.3. 30-day forecast

We trained a more sophisticated RNN that takes in a time window of 100 days and forecasts the exchange rate over the next 30 days. More precisely, the RNN makes a total of 10 predictions for the 1st, 4th, 7th, ..., and 28th days, from which the MSE loss was taken. This design choice was made to make the model more tractable, since we wanted it to learn overall trends in the exchange rate. We also trained a convolutional neural network (CNN) on the same time window to see if it could achieve similar results. At this stage, we had to train our models using Google Colaborator, as the RNN took ∼2.5 hours to train on a CPU.

We also compared training our models on just the USD-EUR exchange rate to training on all 54 time series. The benchmark only fits the USD-EUR exchange rate.

| *Model* | *Average MSE on Testing* |
|---|---|
| Benchmark | $5.44 \times 10^{-4}$ |
| RNN, only USD-EUR | $2.48 \times 10^{-4}$ |
| RNN, all data | $2.61 \times 10^{-4}$ |
| CNN, only USD-EUR | $2.54 \times 10^{-4}$ |
| CNN, all data | $4.15 \times 10^{-4}$ |
| CNN+GRU, only USD-EUR | $2.65 \times 10^{-4}$ |
| CNN+GRU, all data | $3.44 \times 10^{-4}$ |

We see that we were able to achieve better performance with RNNs over CNNs. Another observation was that training on just the exchange rate gave predictions that had lower MSE, but were also flatter, as can be seen in Figure 2 and 3.

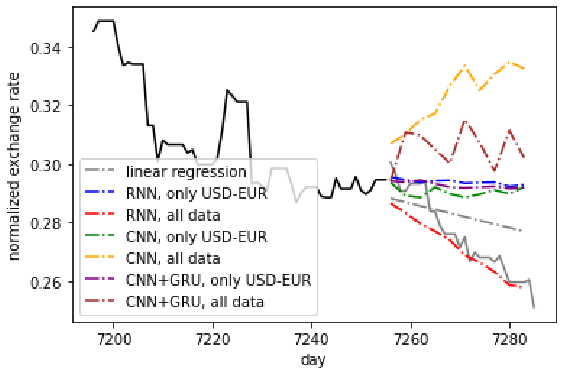

Figure 2. Comparing models for 30-day forecast.

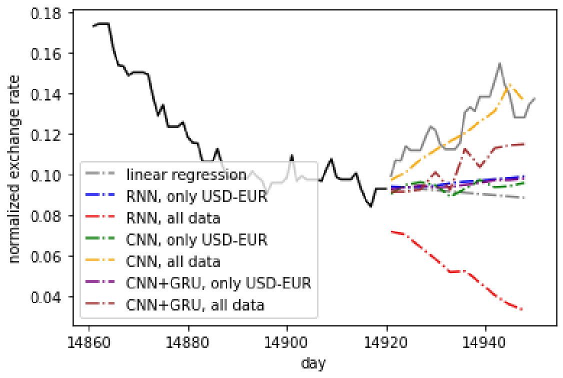

Figure 3. Comparing model for 30-day forecast

Another model we tried is a CNN with inception-modules before a GRU. The inception modules concatenated 1D convolutions with kernel sizes 1,3,5,7, and 9. The rationale is as follows. In price prediction, if the moving average of the slope over a short period is higher than that over a longer time period, that means the price is likely to go up. We want the network to learn features over different timescales, so we included the inception-module. A deep CNN could learn features with different timescale in theory, but is harder to train.

Like with the CNN and RNN, the CNN+GRU network trained with all data, although having a higher MSE than their single-data-trained counterparts, has a more diverse prediction of the trend. On some test sets, it seems to be able to produce the correct slope, as well as the change in volatility of the price, as shown in Figure 3.

### 2.4. 30-day forecast with Fourier Decomposition preprocessing

The task of our next experiment was estimating the benefit of a Fourier Decomposition of the time series as a means of denoising the data. For this experiment, we followed economic orthodoxy and took as our input time series:

– the USD-EUR exchange rate at opening and closing,

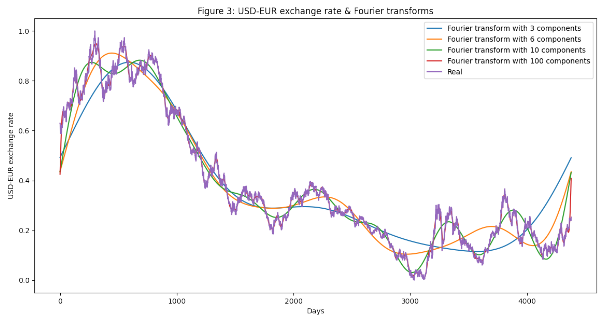

Figure 4. A demonstration of the trade-off between information loss and denoising, for different truncations of the Fourier coefficients. Also notice the bad fitting of FT at the two boundaries.

– the yield of the 9-year bond in Germany,

– the high and low price of oil within the day,

– the US unemployment rate,

– and the effective exchange rate for USD.

These are variables either directly related to the USD-EUR exchange rate, or are indicative of factors such as inflation and the GDP of the underlying economy. For this experiment, we used only post-2000 data and learned on time windows of 30 days. Like before, our benchmark model is Linear Regression which gives a mean MSE of $4 \times 10^{-4}$. With these features, it becomes extremely hard to train stacked GRU or LSTM models that beat Linear Regression. For this reason, we were led to the following two decisions regarding model-architecture: choose a single GRU or LSTM layer with many hidden units, and use some Fourier Transform (FT) preprocessing to denoise the signal. We pass the 30-day history subsequences for every feature through the FT and keep only the 10 leading Fourier components. We decided not to use FT on the 30-day prediction range, or even on the whole 60-day range, since in the former case we would be grappling with the well-known problem of bad fitting of FT on boundaries, and in the latter our GRU would just be trying to learn sines and not economic forecasting.

The results of our experiment with denoising are summarized in the table below. Since LSTMs are well known to be harder to train, to achieve a fair comparison we picked the GRU with the best validation loss among 10 different learning rates, and similarly for the LSTM. We also included an ensemble prediction based on the mean of the GRU and L. R. predictions. Ensemble learning is known to improve performance even from weak learners by cancelling error in opposite directions.

| *Model* | *test MSE* |
|---|---|
| Benchmark | $4.3 \times 10^{-4}$ |
| GRU | $3.7 \times 10^{-4}$ |
| LSTM | $7.8 \times 10^{-4}$ |
| Ensemble | $3.0 \times 10^{-4}$ |

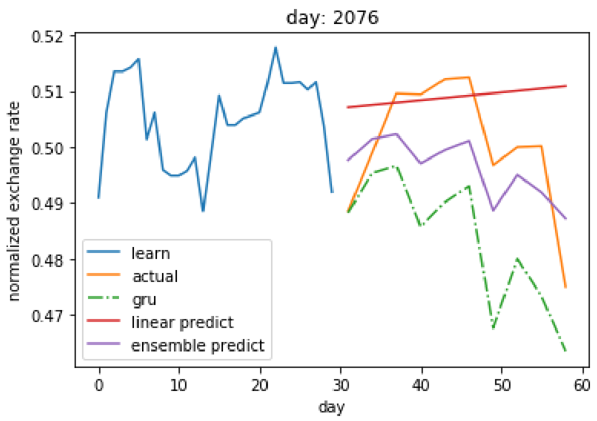

Figure 5. Forecast example with Fourier preprocessing

## 3. Discussion

For the 30-day forecast, we were able to produce models that beat linear regression, with up to a 55% reduction in the average MSE. The models with the lowest MSEs trained on just the USD-EUR exchange rate and made flat predictions. This leads us to believe that these models are minimizing their losses by making "vanilla" predictions, i.e. a straight line with small slope, to avoid being penalized for bad predictions. Other models that trained on the entire data set had slightly worse performance, but were able to produce concrete predictions for the long-term trend. For someone interested in making informed decisions when trading currencies, these models are more useful.

One might suspect that the reason why adding more features results in worse predictions is due to under-fitting, similar to the kind that plagues deep networks. We speculate the maximum amount of information one can extract from the USD-EUR exchange rate by itself is limited. For greater predicting power, it may be necessary to use feature selection to include just the most relevant time series and/or utilizing additional time series in a more sophisticated way other than naively appending them as features.

For the 30-day forecast with Fourier denoising, which used a different test set than the first experiment, we observed that Fourier denoising facilitated the training of big NNs. We were able to beat the benchmark non-NN model, and ensemble learning resulted in a 35% improvement over Linear Regression. In fact, as we see on Fig. 5, the GRU has learned to make highly sophisticated and diverse predictions based on the history window. We see that based on the preceding history, it predicts different slopes, increases or decreases, and different number of "knees."

In conclusion, all experiments are compatible with the hypothesis of the project, that RNN algorithms can outperform non-NN algorithms in forecasting economic timeseries.

## References

[1] Global financial data. finaeon.globalfinancialdata.com, 2021.

[2] J. Hwang. Modeling financial time series using lstm with trainable initial hidden states, 2020.

[3] D. Tlegenova. Forecasting exchange rates using time series analysis: The sample of the currency of kazakhstan, 2015.

## A. Model Architectures

### A.1. Next-day predictions

The training set contains 15 747 overlapping windows of 30 data points. The validation and testing sets each contain 50 non-overlapping windows.

*Linear regression*: Fit on $k = 5$ points.

*GRU*:

| |
|---|
| GRU, 100 units |
| Dense, 1 unit |

Optimized with Adam and learning rate 1e-3, decaying 0.9 per epoch. Trained over 30 epochs. Kept model with lowest MSE loss on validation set.

*LSTM*:

| |
|---|
| LSTM, 100 units |
| Dense, 1 unit |

Same as GRU, but trained over 50 epochs.

### A.2. 30-day predictions

The training set contains 10 759 overlapping windows of 100 data points. The validation and testing sets each contain 50 non-overlapping windows.

*Linear regression*: Fit on $k = 30$ points.

*RNN*:

| |
|---|
| GRU, 100 units |
| Dropout, 20% |
| GRU, 100 units |
| Dropout, 20% |
| GRU, 100 units |
| Dense, 10 units |

Optimized with Adam and learning rate 1e-3, decaying 0.9 per epoch. Trained over 30 epochs. Kept model with lowest MSE loss on validation set.

*CNN*:

| |
|---|
| Conv, 32 filters, size 3, stride 1 |
| Conv, 64 filters, size 3, stride 1 |
| Average pool, size 2, stride 2 |
| Conv, 128 filters, size 3, stride 1 |
| Average pool, size 2, stride 2 |
| Dense, 100 units |
| Dense, 10 units |

Optimized with Adam and learning rate 1e-3, decaying 0.9 per epoch. Trained over 30 epochs. Kept model with lowest MSE loss on validation set.

*CNN+GRU:*

| | |
|---|---|
| Concatenate | Conv, 20 filters, size 1, stride 1<br>Conv, 20 filters, size 3, stride 1<br>Conv, 20 filters, size 5, stride 1<br>Conv, 20 filters, size 7, stride 1<br>Conv, 20 filters, size 9, stride 1 |
| Average Pool, size 2, stride 2 | |
| Concatenate | Conv, 100 filters, size 1, stride 1<br>Conv, 80 filters, size 3, stride 1<br>Conv, 60 filters, size 5, stride 1<br>Conv, 40 filters, size 7, stride 1<br>Conv, 20 filters, size 9, stride 1 |
| GRU, 200 units | |
| Dense, 10 units | |

When trained on single time series, the first convolutional layer with filter size 1 is omitted. Optimized with Adam and learning rate 1e-4, decaying 0.9 per epoch. Recurrent dropout and dropout of 0.1 at GRU layer. Trained over 35 epochs. Kept model with lowest MSE loss on validation set.

### A.3. 30-day w/ Fourier preprocessing

Our train, validation and test sets were roughly the years [2000, 2011], [2012, 2015] and [2016, 2020], with sufficient gaps between them to prevent correlations between these sets. The training set contains 4384 windows of 30 data points. The validation set contains 1460 windows and the testing set contains 1825 windows.

*Linear regression*: Fit on $k = 30$ points.

*LSTM*:

| |
|---|
| Fourier Denoising with 10 leading components |
| LSTM, 1000 units and L1 kernel-regularization |
| Dropout, 30% |
| Dense, 10 units |

Optimized with Adam and learning rate 5e-5, with early stopping based on the validation loss with 20 epochs of patience, and maximum number of epochs 300.

*GRU*:

| |
|---|
| Fourier Denoising with 10 leading components |
| GRU, 1000 units and L1 kernel-regularization |
| Dropout, 30% |
| Dense, 10 units |

Optimized with Adam and learning rate 1e-5, with early stopping based on the validation loss with 20 epochs of patience, and maximum number of epochs 300.

*Ensemble Learner*: Predicts the average of the predictions of the GRU and the Linear Regression.