Exploring Recurrent Neural Network Frameworks — A Case Study on Foreign Exchange Rate Forecasting

Alexander Jakob Dautel

Outline

1. Recurrent Neural Networks
2. Related Work
3. Experiment
4. Conclusion
5. References
6. Appendix

Recurrent Neural Network Variants

- Simple Recurrent Neural Networks (SRNN), 1980s
- Long Short-Term Memory (LSTM), 1997
- Gated Recurrent Units (GRUs), 2015

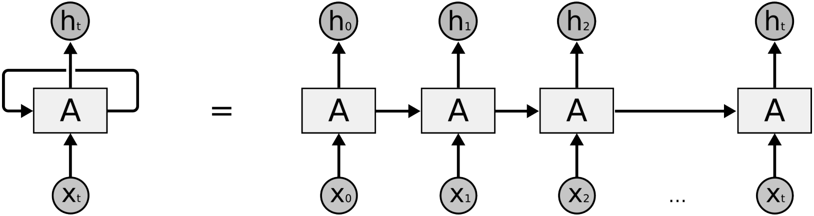

Figure 1: Unrolled RNN

Gated Recurrent Neural Networks

- Gates control cell state or hidden state
- Better gradient flow allows long-term information
- Memory cell $\widehat{=}$ complex activation function

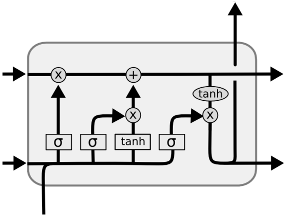

Figure 2: LSTM Cell

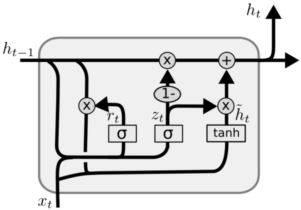

Figure 3: GRU

Financial Forecasting with RNNs (Selection)

Authors	Year	Data	RNN	Benchmarks
Kamijo and Tanigawa	1990	Chart Signals	SRNN	-
Tenti	1996	Forex	SRNN	-
Giles et al.	2001	Forex	SRNN	FNN
Xiong et al.	2015	Index (Vola)	LSTM	L1, L2, GARCH
Fischer and Krauss	2018	Stocks	LSTM	FNN, RAF, LOG
Shen et al.	2018	Indices	GRU	FNN, SVM

Table 1: Selected scientific studies empoying RNNs for financial forecasting.

Foreign Exchange Rates

EUR, GBP, JPY, and CHF vs. USD 1971-2017

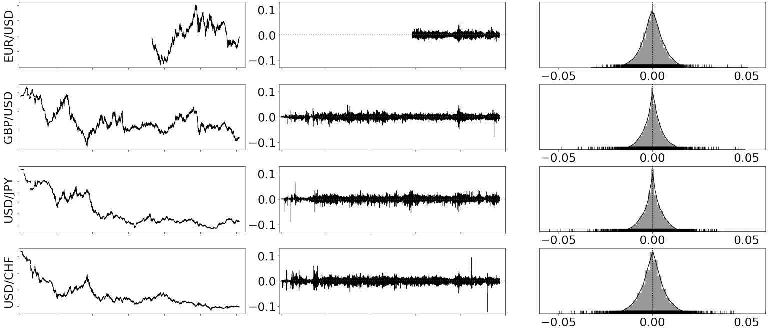

Figure 4: Prices, one-day percentage returns, KDE/rug plots.

Data Preprocessing

- Percentage returns
- Min-max-scaling to $[-1, 1]$[1]
- Supervised learning problem
  - Feature: windows of 240 observations
  - Targets: observation following each feature window, binary

[1] Scaler fitted to training data only.

Prediction Set-up

- 43 overlapping study periods S (15 for EUR/USD)
- $\forall S$: 750 obs. training & validation, 250 obs. trading

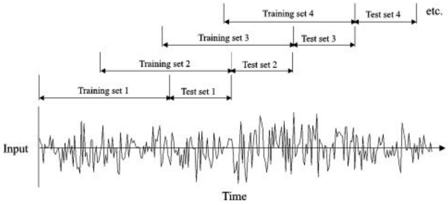

Figure 5: Rolling walk forward prediction windows.

Models

- FNN, SRNN, LSTM, GRU:

```
OPERATION           DATA DIMENSIONS   WEIGHTS(N)   WEIGHTS(%)

    Input   #####    240    1
     LSTM   LLLLL -------------------     10400     20.5%
     tanh   #####    240   50
  Dropout    | || -------------------         0      0.0%
            #####    240   50
     LSTM   LLLLL -------------------     20200     39.7%
     tanh   #####    240   50
  Dropout    | || -------------------         0      0.0%
            #####    240   50
     LSTM   LLLLL -------------------     20200     39.7%
     tanh   #####          50
  Dropout    | || -------------------         0      0.0%
            #####          50
    Dense   XXXXX -------------------        51      0.1%
  sigmoid   #####           1
```

Figure 6: Neural network model topology (example: LSTM).

- Naive benchmark: $\hat{y}_t = y_{t-1}$

Results

Discrepancy between loss function and trading strategy.

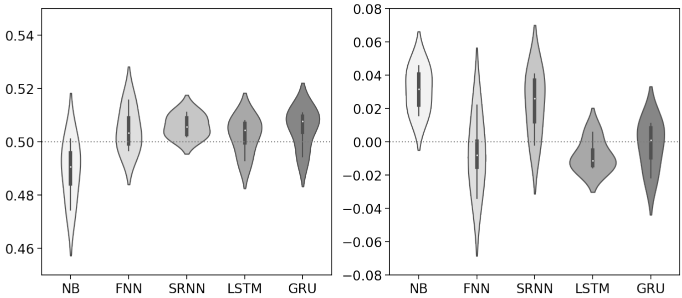

Figure 7: Accuracy vs. simple trading strategy returns.

Reflection

- Couple loss function and trading strategy: maximise profit or Sharpe ratio during training
- Financial markets: non-stationarity, predictability?
- Hyperparameter tuning: 576 individual models[2] and large hyperparameter space[3]
- Low-confidence predictions

[2] Four time series, {15, 43, 43, 43} study periods, four models.
[3] Variations of input features, network topology, training parameters.

Outlook

- For now:
  - LSTM and GRUs state-of-the art in many fields.
  - Require meticulous tuning and experimenting for success in financial markets.
- What's next?
  - CNNs for time series forecasting
  - Neural Turing Machines
  - Attention-based algorithms

Thank you!

"It is perfectly true, what machine learners say: that neural networks must be trained backwards. But they forget the other proposition: that they must be applied forwards."[4]

Figure 8: Søren Kierkegaard, backpropagation visionary.

[4] Not a true quote.

Literature (Selection)

- K. Kamijo and T. Tanigawa (1990)
  Stock price pattern recognition—a recurrent neural network approach
  1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, vol. 1 , pp. 215–221
- P. Tenti (1996)
  Forecasting foreign exchange rates using recurrent neural networks
  Applied Artificial Intelligence, vol. 10, no. 6, pp. 567–582

Literature (Selection)

C. L. Giles, S. Lawrence, and A. C. Tsoi (2001)
Noisy Time Series Prediction using Recurrent Neural Networks and Grammatical Inference
Machine Learning, vol. 44, no. 1, pp. 161–183

R. Xiong, E. P. Nichols, and Y. Shen (2015)
Deep Learning Stock Volatility with Google Domestic Trends
arXiv:1512.04916 [q-fin]

Literature (Selection)

T. Fischer and C. Krauss (2018)
Deep learning with long short-term memory networks for financial market predictions
European Journal of Operational Research, vol. 270, no. 2, pp. 654–669

G. Shen, Q. Tan, H. Zhang, P. Zeng, and J. Xu (2018)
Deep Learning with Gated Recurrent Unit Networks for Financial Sequence Predictions
Procedia Computer Science, vol. 131, pp. 895–903

Data Overview

	EUR/USD	GBP/USD	USD/JPY	USD/CHF
Observations	4687	11708	11702	11708
Mean	0.0000	-0.0000	-0.0001	-0.0001
Standard Dev.	0.0063	0.0060	0.0065	0.0073
Minimum	-0.0296	-0.0784	-0.0907	-0.1221
25 % Quantile	-0.0034	-0.0029	-0.0030	-0.0038
Median	0.0000	0.0001	0.0000	0.0000
75 % Quantile	0.0035	0.0029	0.0031	0.0036
Maximum	0.0473	0.0470	0.0646	0.0930
Skewness	0.1511	-0.3216	-0.5540	-0.2305
Kurtosis	2.2591	6.9514	8.6128	12.3697

Table 2: Statistical properties of the one-day percentage returns of selected currencies.

Data Preprocessing

- Percentage returns r_t^c
- Min-max-scaling to $[-1, 1]$:[5] $\tilde{r}_t^c$
- Supervised learning problem
  - Feature: windows of 240 observations

$$X_t = \{\tilde{r}^c_{t-240}, \tilde{r}^c_{t-240+1}, \tilde{r}^c_{t-240+2}, \ldots, \tilde{r}^c_{t-1}\}$$

  - Targets: observation following each feature window, binary

$$y_t^c = \begin{cases} 1 & \text{if } r_t^c \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

[5] Scaler fitted to training data only.

Parameter Spaces for Hyperparameter Tuning

- Number of hidden layers: 1, 2, 3, 4
- Number of neurons per hidden layer: 25, 50, 100, 200, 400, 800, 1600
- Dropout: 0 to 60 percent, in steps of 10 percent
- Optimizer and learning rate: `Adam` and `RMSprop` with various learning rates
- Batch size: 16, 32, 64, 128, 256

Model Architectures

- Three hidden layers
- 50 neurons per hidden layer
- 25 percent dropout after each hidden layer
- Activation functions: `relu` (FNN), `tanh` (RNNs), `sigmoid` (LSTM & GRU gates)
- Output activation functions: `sigmoid`

Training Parameters

- Loss function: binary cross-entropy[6]

$$L_S(y_{T_S}, \hat{y}_{T_S}) = -\frac{1}{|T_S|} \sum_{t \in T_S} (y_t \log(\hat{y}_t) + (1 - y_t) \log(1 - \hat{y}_t))$$

- Regularization: dropout, 20 percent hold-out data, early stopping

[6] L_S simplifies to $L_S(y_{T_S}, \hat{y}_{T_S}) = -\frac{1}{|T_S|} \sum_{t \in T_S} \log(\hat{y}_t)$ in the binary classification task with labels (0,1).

Simple Trading Strategy

- Trade every prediction:

$$\tilde{r}_t^c = \begin{cases} r_t^c & \text{if } \hat{y}_t \geq 0.5 \\ -r_t^c & \text{otherwise} \end{cases}$$

- Clear position after one day
- Annualized net returns:

$$R_S = \prod_{t \in T_S} (1 + \tilde{r}_t^c) - 1$$

Results in Numbers

Model	Log Loss	Acc.	AUC	Returns	SD	SR
NB	-	0.4921	0.4880	0.0307	0.0064	0.0161
FNN	0.7026	0.5062	0.5061	-0.0126	0.0064	-0.0071
SRNN	0.7115	0.5103	0.5073	0.0195	0.0064	0.0090
LSTM	0.6993	0.5076	0.5088	-0.0072	0.0064	-0.0050
GRU	0.6992	0.5107	0.5085	0.0014	0.0057	0.0024

Table 3: Results across all time series (weighted by number of study periods).