Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions: A Brief Review

Frances Lin

May 2025

Background

Main Paper:

- Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara. 2021. "Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions." International Journal of Forecasting 37 (1): 388–427.

Resource:

- Olah, Christopher. 2015. "Understanding LSTM Networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Background

- Time series forecasting has been traditionally a topic in Statistics and Econometrics, from simple methods such as Seasonal Naive and Simple Exponential Smoothing, to more complex ones such as ETS and ARIMA.
- However, such traditional univariate techniques lack key features needed for advanced forecasting tasks.
  - **Comment.** There are even more complex classes of models, such as state space models (SSMs).
- With the ever increasing availability of data, ANNs (Artificial Neural Networks) have become a dominant and popular technique for machine learning tasks.

RNN

- RNNs (Recurrent Neural Networks) are the most commonly used neural networks architecture for sequence prediction problems.
- An RNN consists of multiple RNN units. Commonly used units for sequence modelling tasks are:
  - the Elman RNN cell, the Long Short-Term Memory (LSTM) cell and the Gated Recurrent Unit (GRU).
- Although most commonly used for natural language processing (NLP) tasks, these architectures are used in various time series forecasting tasks as well.

RNN

- An RNN is composed of repeating cells or units that unfold or unroll over time,
  - where each cell passes recurrent information stored in the hidden state from one time step to the next.

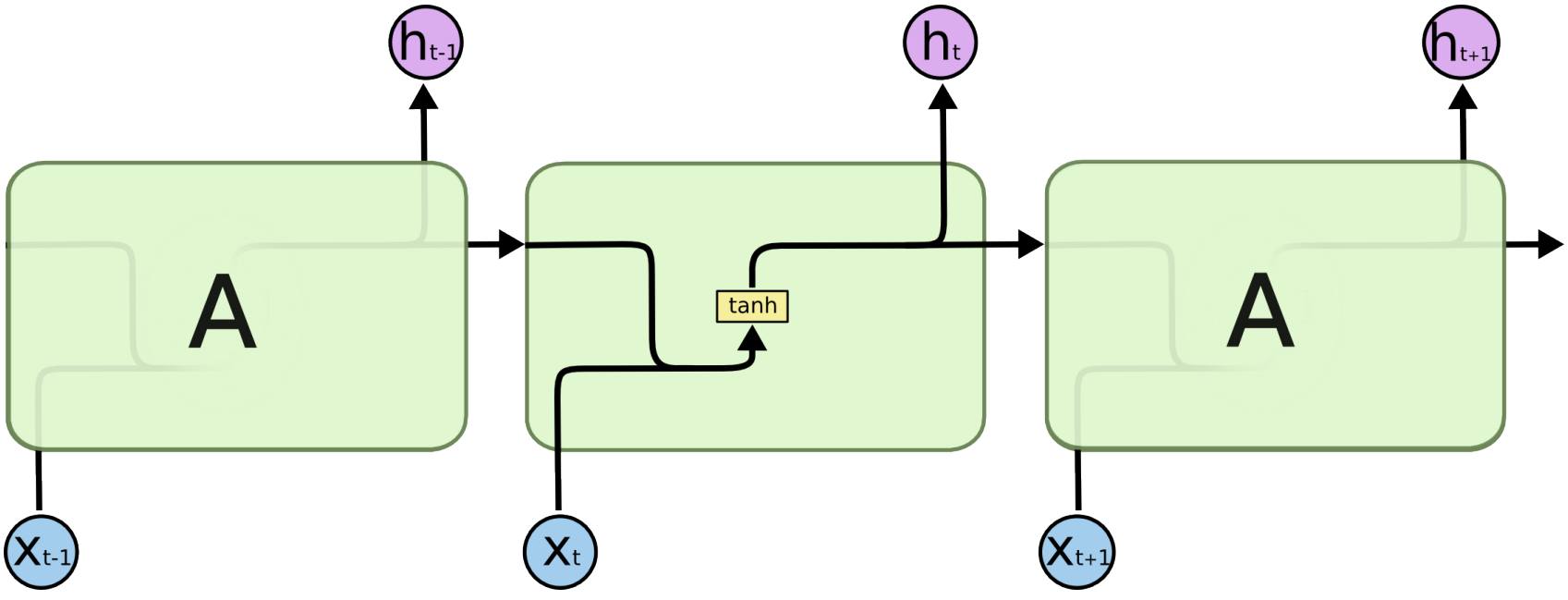


Figure 1: Unfolded Version of RNN. Reproduced from Olah (2015).

Recurrent Units

Types of RNN Cells are:

- Elman Recurrent Unit (ERNN)
- Long Short-Term Memory (LSTM)
  - We focus on the LSTM with a forget gate, as this is the version implemented in `PyTorch`.
  - Several LSTM variants exist, but their performance is generally comparable.
- Gated Recurrent Unit (GRU)

Elman Recurrent Unit

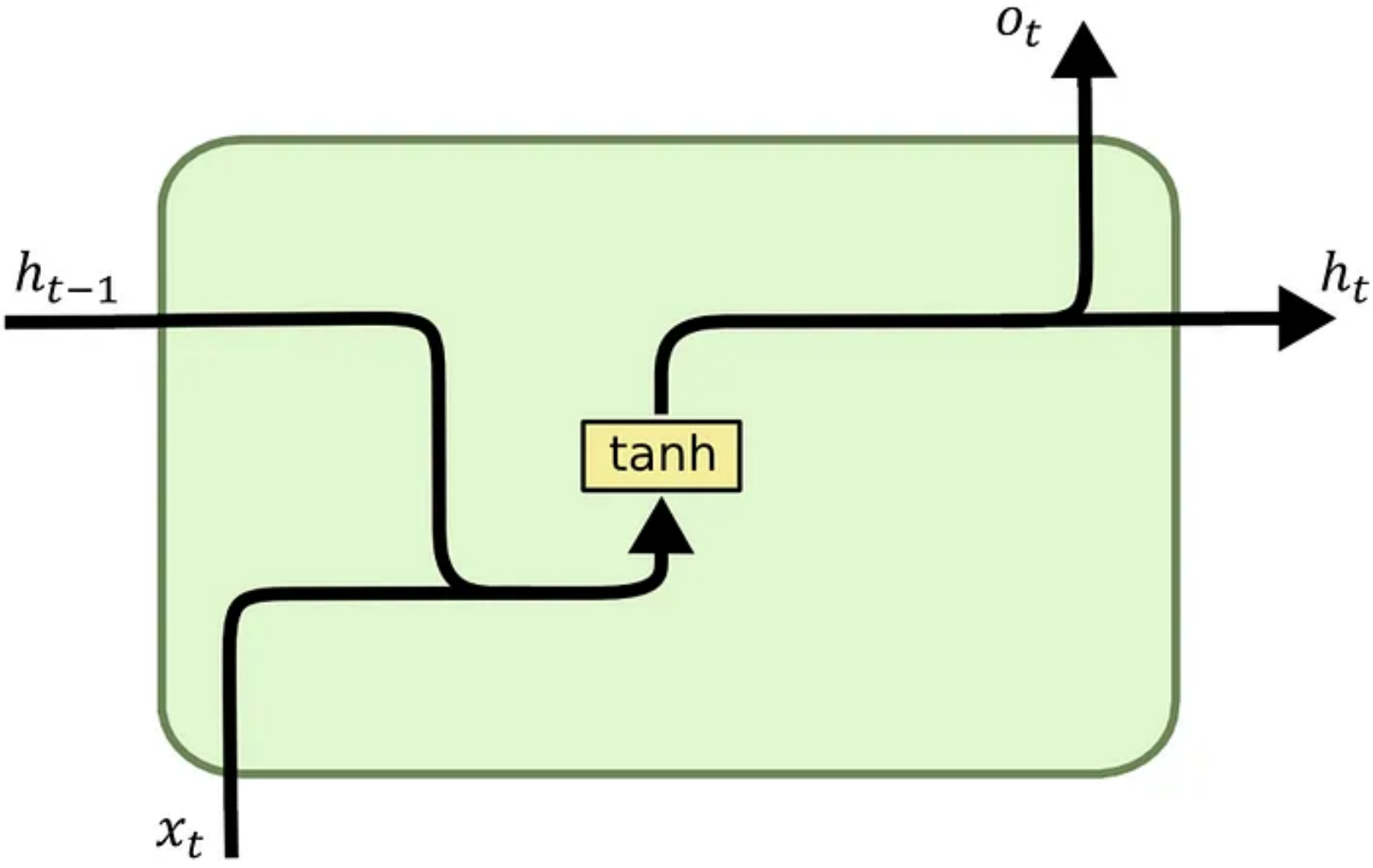


Figure 2: Folded Version of RNN. Reproduced from Olah (2015).

Elman Recurrent Unit

Let x_t, h_t, and y_t denote the input data, the hidden state, and the output at time t, respectively. Then, an RNN unit can be expressed as:

$$\begin{aligned} h_t &= \sigma(W_{ih} h_{t-1} + W_{ix} x_t + b_i), \\ o_t &= tanh(W_{oh} \cdot h_t + b_o), \end{aligned} \tag{1}$$

- W denote the weight matrices, and b the bias vectors, where subscripts i and o refer to the input and the ouput steps.
- σ and $tanh$ are the activation functions, the logistic sigmoid function and the hyperbolic tangent function, respectively.

Elman Recurrent Unit

- The ERNN cell suffers from the well-known vanishing and exploding gradient problems when trained with Backpropagation Through Time (BPTT) over long sequences.
- The LSTM unit was proposed to help capture long-term dependencies in sequence data and alleviate the vanishing gradient problem.
  - LSTM alleviates this issue to some extent.
  - **Comment.** This is one reason why transformer-based model is increasingly applied to forecasting tasks, but it has its limitations too.

LSTM

- An LSTM unit extends an RNN by introducing a cell state and three gates.
  - The cell state carries long-term dependencies. The hidden state encodes short-term patterns.
  - The gates (i.e., the forget, input, and output gates) regulate the flow of information.

LSTM

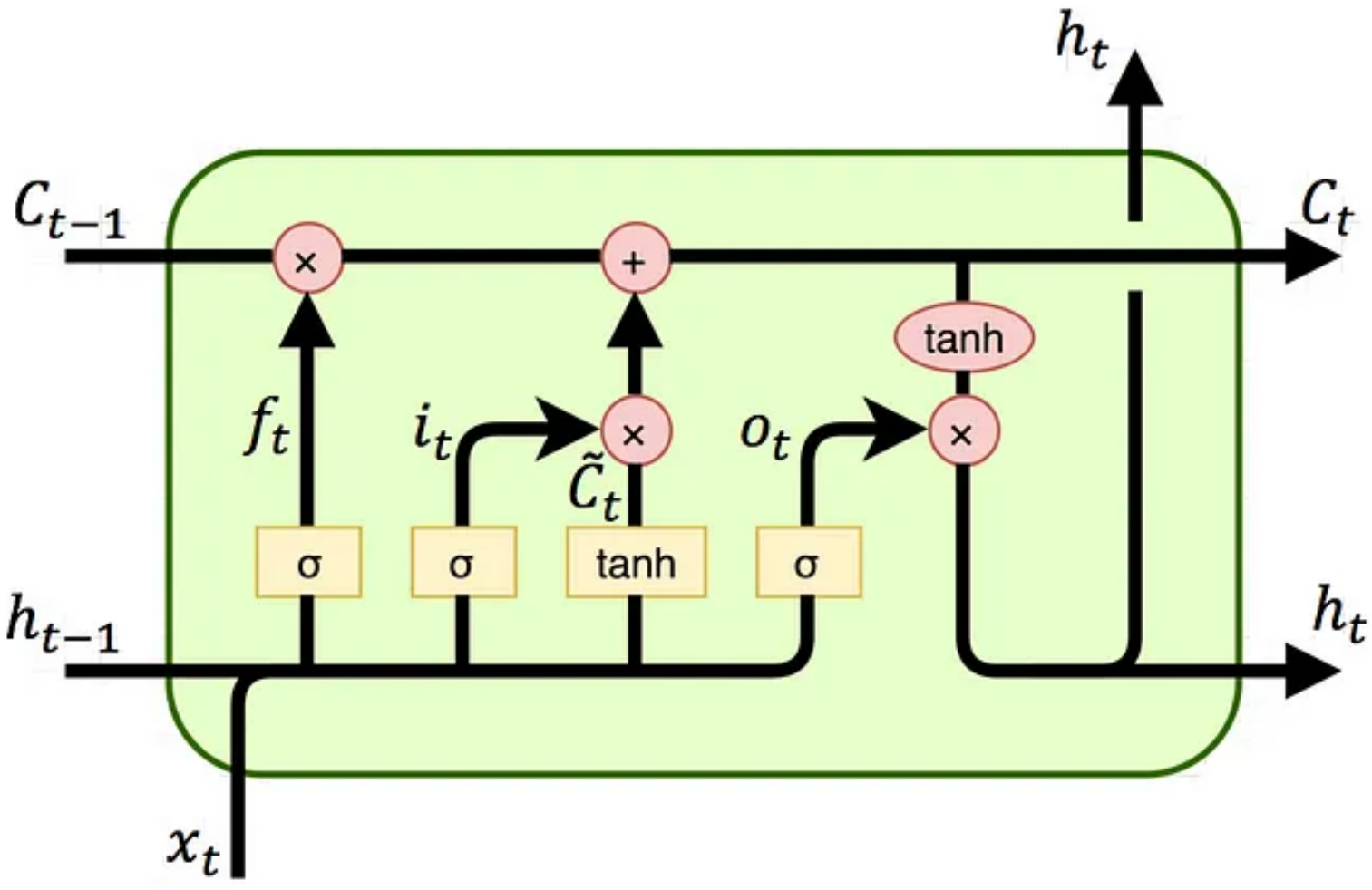


Figure 3: Folded Version of LSTM. Reproduced from Olah (2015).

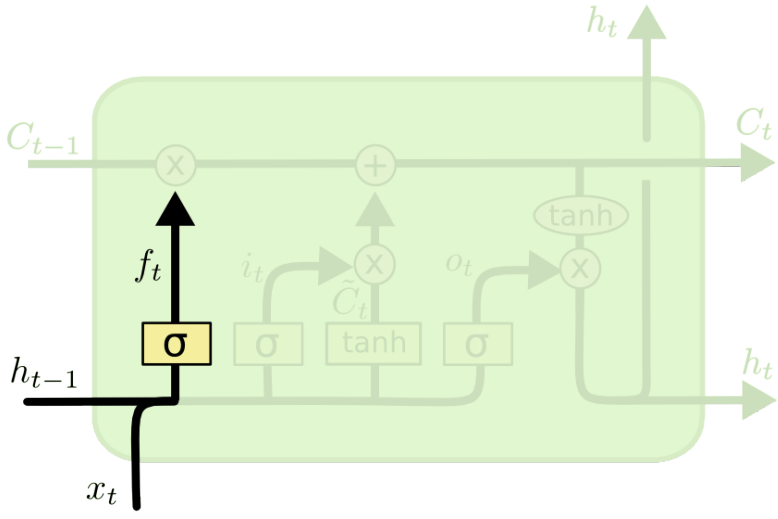
LSTM

Let f_t, i_t, and o_t represent the forget, the input, and the output gate vector at time t, respectively. Then, an LSTM unit can be expressed as:

$$
\begin{aligned}
f_t &= \sigma(W_{fh} h_{t-1} + W_{fx} x_t + b_f), \\
i_t &= \sigma(W_{ih} h_{t-1} + W_{ix} x_t + b_i), \\
\tilde{c}_t &= \tanh(W_{\tilde{c}h} h_{t-1} + W_{\tilde{c}x} x_t + b_{\tilde{c}}), \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t, \\
o_t &= \sigma(W_{oh} h_{t-1} + W_{ox} x_t + b_o), \\
h_t &= o_t \cdot \tanh(c_t).
\end{aligned}
\tag{2}
$$

- W denote the weight matrices, b the bias vectors,
- σ and *tanh* the logistic sigmoid function and the hyperbolic tangent function, respectively.

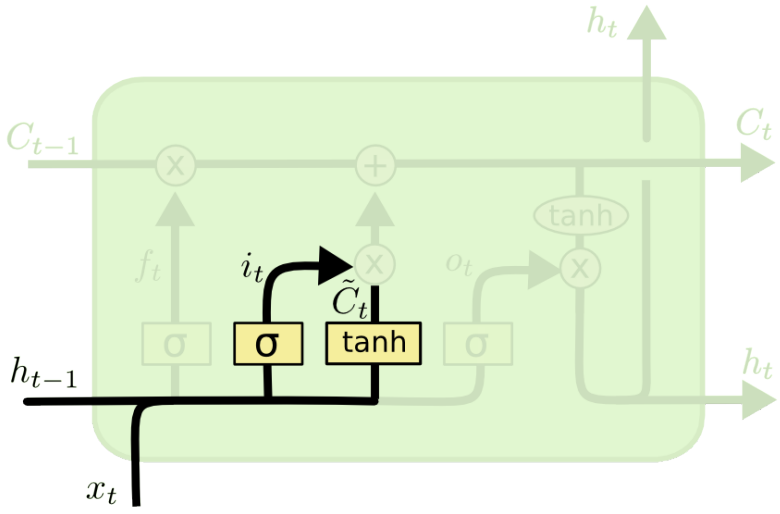
LSTM Step 1



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- The forget gate, f_t,
  - controls the extent to which information to discard or retain.
  - outputs values in range $[0, 1]$, where 0 means the information is completely discarded, and 1 means it is fully retained.

LSTM Step 2

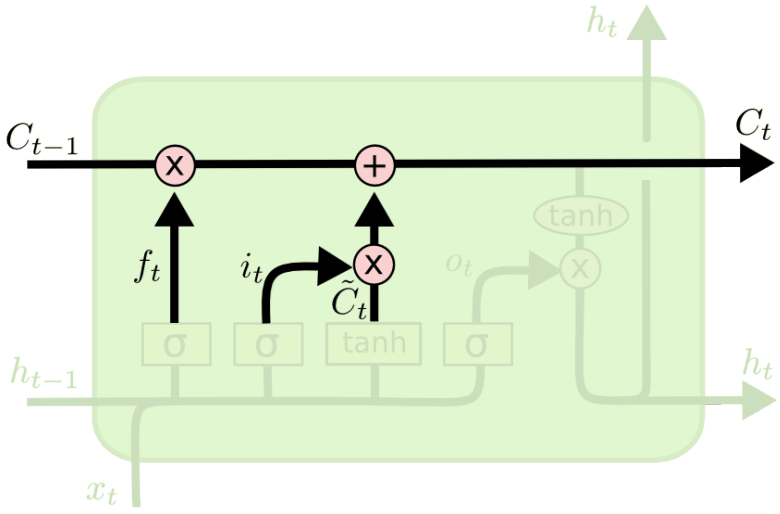


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

- The input gate, i_t,
  - regulates the amount of new information to add or discard.
  - also outputs values in in range $[0, 1]$, where 0 means no information is added, and 1 means it is fully added.
- The network computes the candidate values, $\tilde{c}$, which represents the proposed new information.

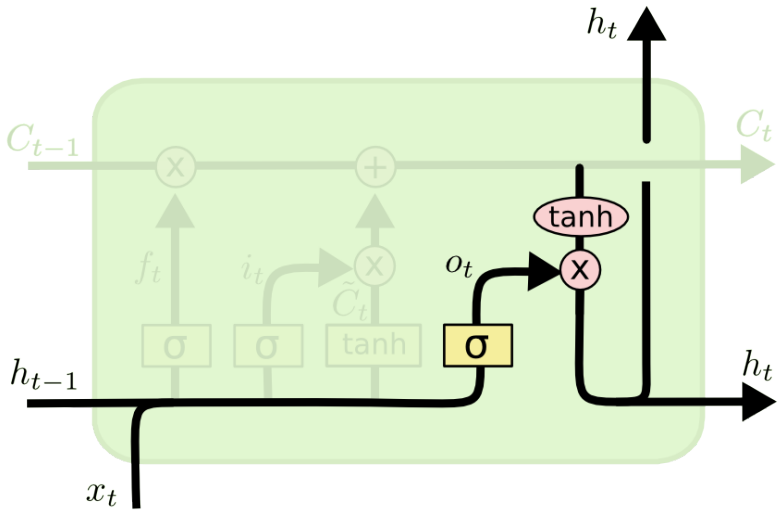
LSTM Step 3



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Next, the cell state, c_t, is updated by combining the previous cell state, c_{t-1}, and the candidate values, $\tilde{c}$.
- Recall that f_t controls how much irrelevant information to discard, while i_t determines how much new information to add when updating the cell state.

LSTM Step 4

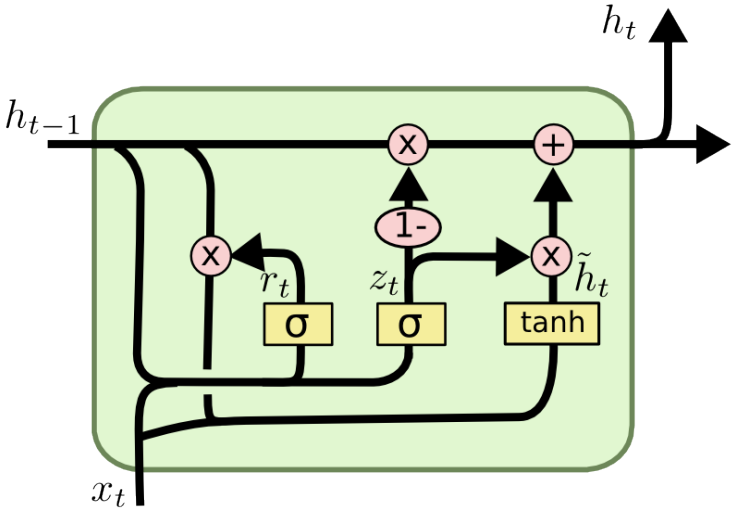


$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

- Then, the output gate, o_t, determines the extent to which the cell state, c_t, is exposed to the hidden state, h_t.
- The hidden state, h_t, is updated by taking the cell state, c_t, and scaling it with the output gate, o_t.

GRU



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

- GRU has gained popularity for its simpler design.
  - Replace forget and input gates with reset gate.
  - Merge the cell state and hidden state.

Experimental Setup

The datasets used for the experiments are taken from the following forecasting competitions.

- CIF 2016 Forecasting Competition Dataset
- M3 Forecasting Competition Dataset
- M4 Forecasting Competition Dataset
- NN5 Forecasting Competition Dataset
- Wikipedia Web Traffic Time Series Forecasting Competition Dataset
- Tourism Forecasting Competition Dataset

Experimental Setup

See the paper for details on other steps such as data splitting, trend normalization, and mean-variance normalization, as well as hyperparameter tuning and performance evaluations.

Results

- On the CIF, M3, M4, and Tourism datasets, no model is significantly better than the two benchmarks (i.e., ETS and ARIMA).
- On the NN5 dataset, the RNN models have only managed to perform significantly better than ARIMA, but not ETS.
- Across all the datasets, it can be seen that the RNN models which perform significantly better than the benchmarks are mostly the variants of the Stacked architecture.
  - **Comment.** See Table 5: Mean SMAPE Results for details, if needed. But a better plot (i.e., Figure 20) is coming up.

Figure 13: Relative Performance of Different Recurrrent Cell Types

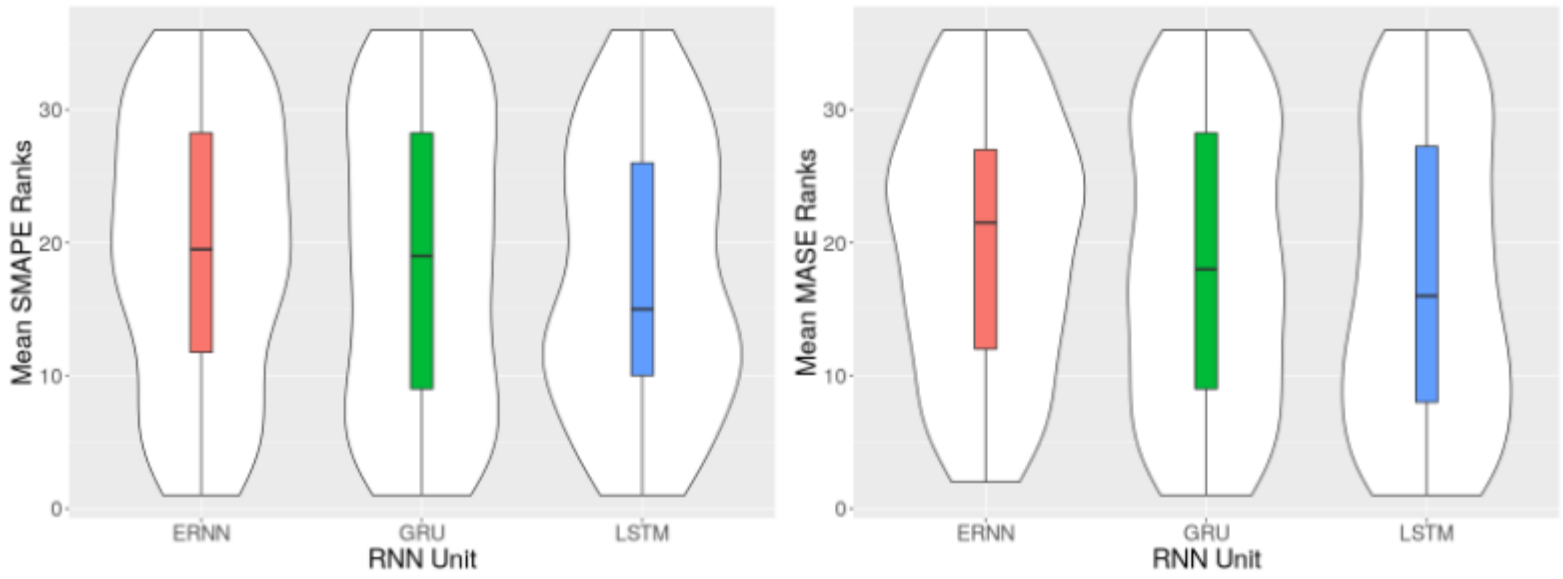


Figure 13: Relative Performance of Different Recurrrent Cell Types

- The difference is not significant based on the p-value.

Figure 14: Relative Performance of Different Optimizers

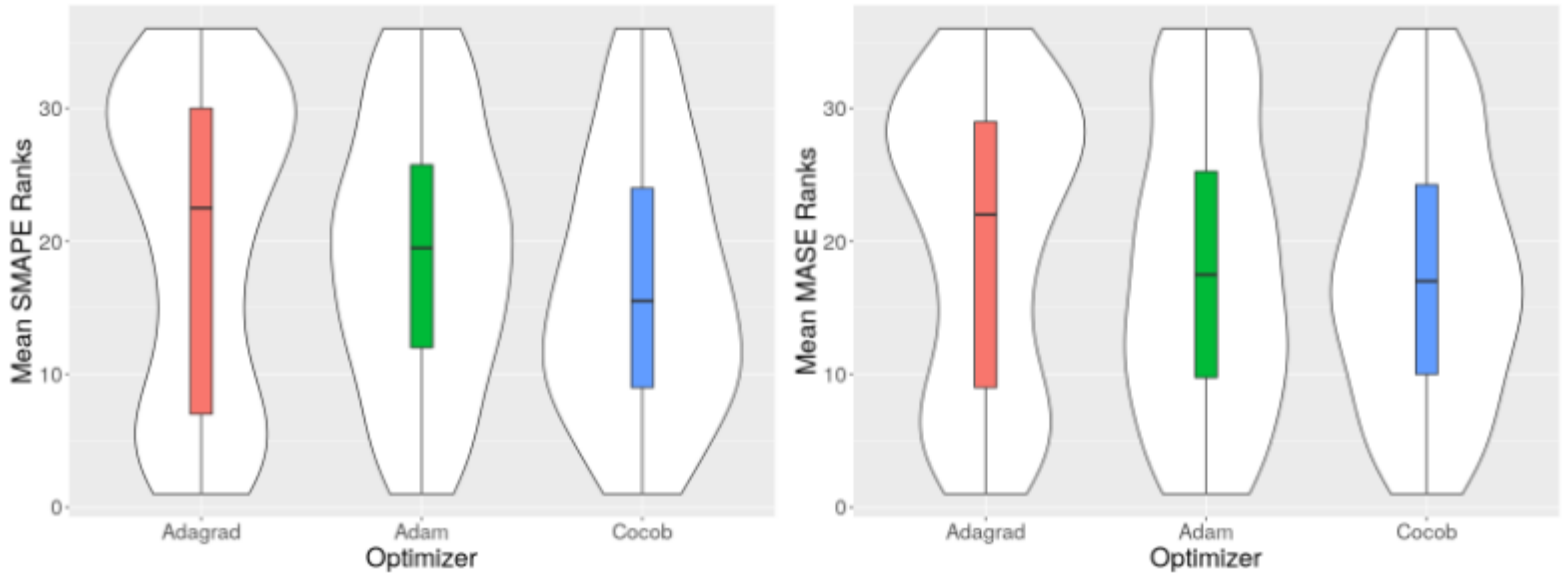
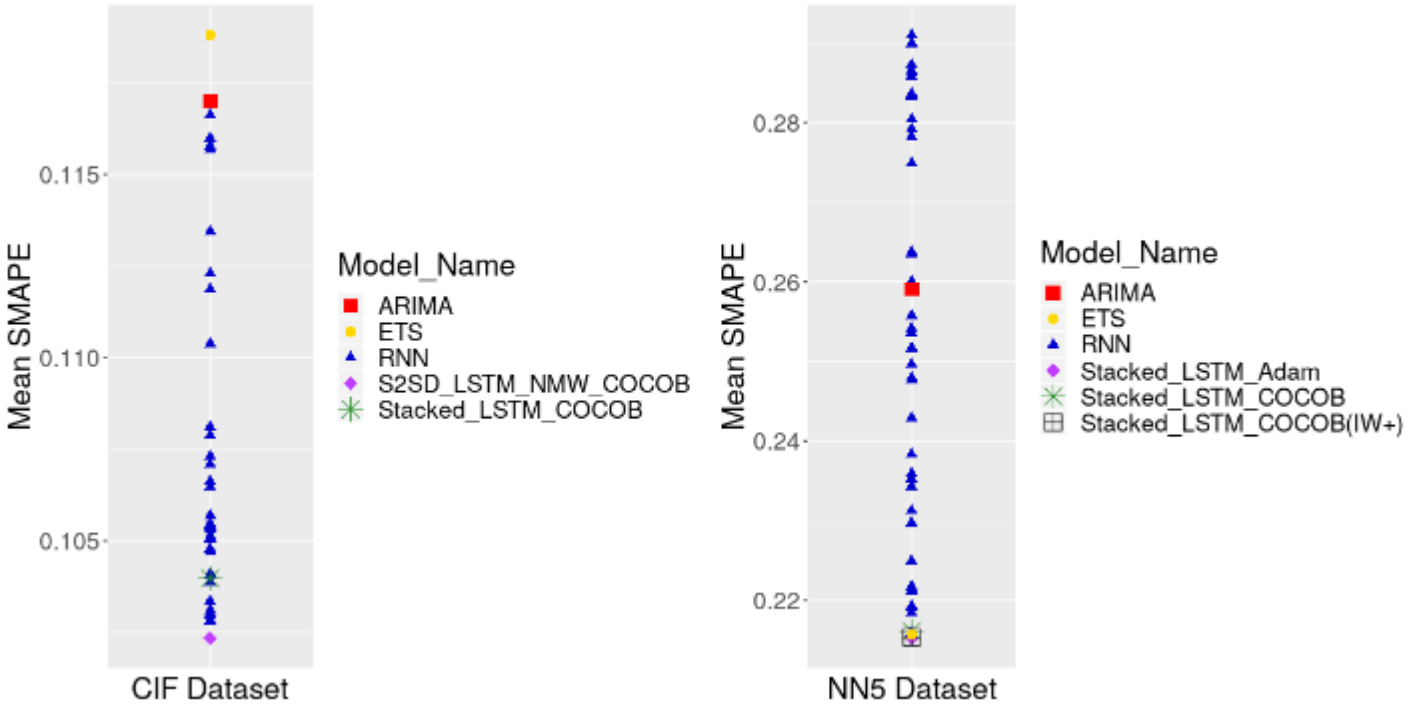


Figure 14: Relative Performance of Different Optimizers

- The difference is not significant based on the p-value.

Figure 20: Performance of RNNs Compared to Traditional Univariate Techniques (1/3)



- An RNN architecture is able to outperform the benchmark techniques on all the datasets except the M4 dataset.
- **Typo.** ETS has the lowest mean SMAPE for the Tourism datatset.

Figure 20: Performance of RNNs Compared to Traditional Univariate Techniques (2/3)

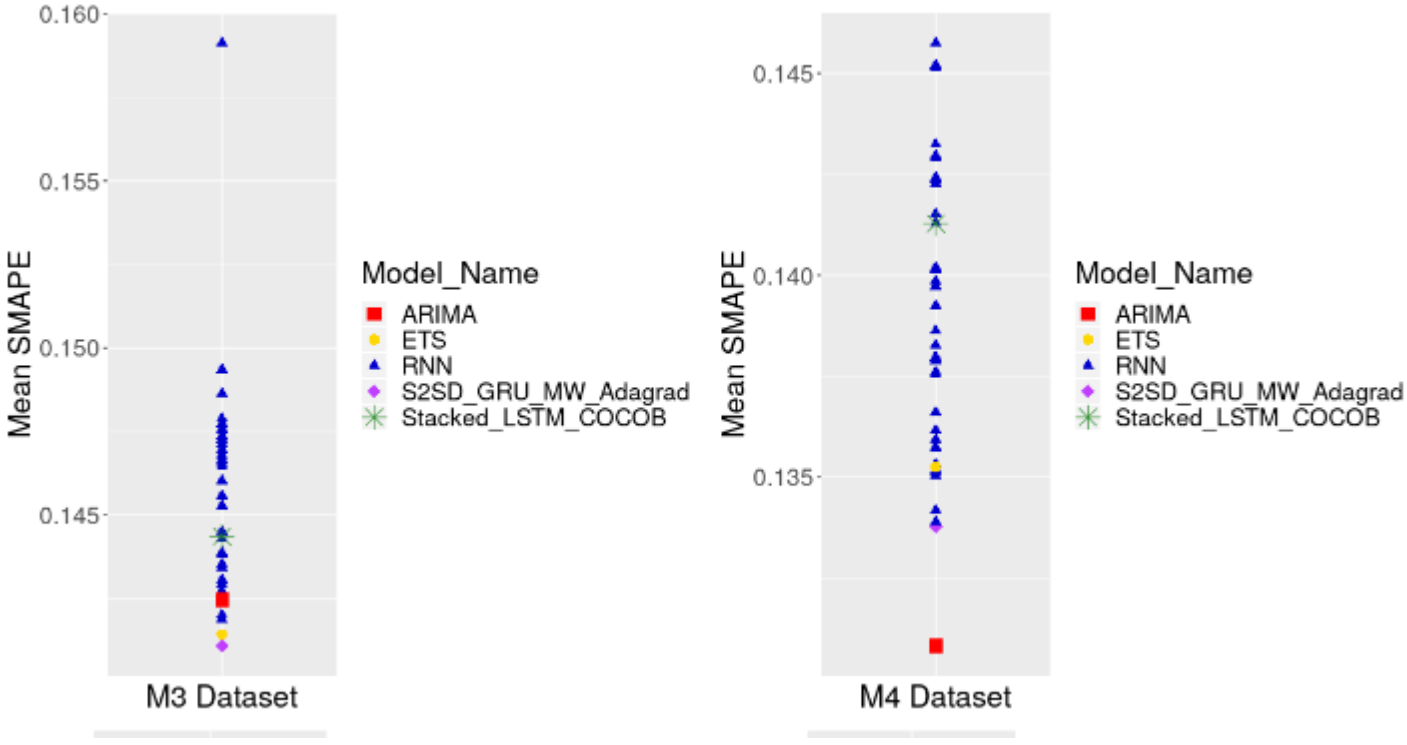


Figure 20: Performance of RNNs Compared to Traditional Univariate Techniques (3/3)

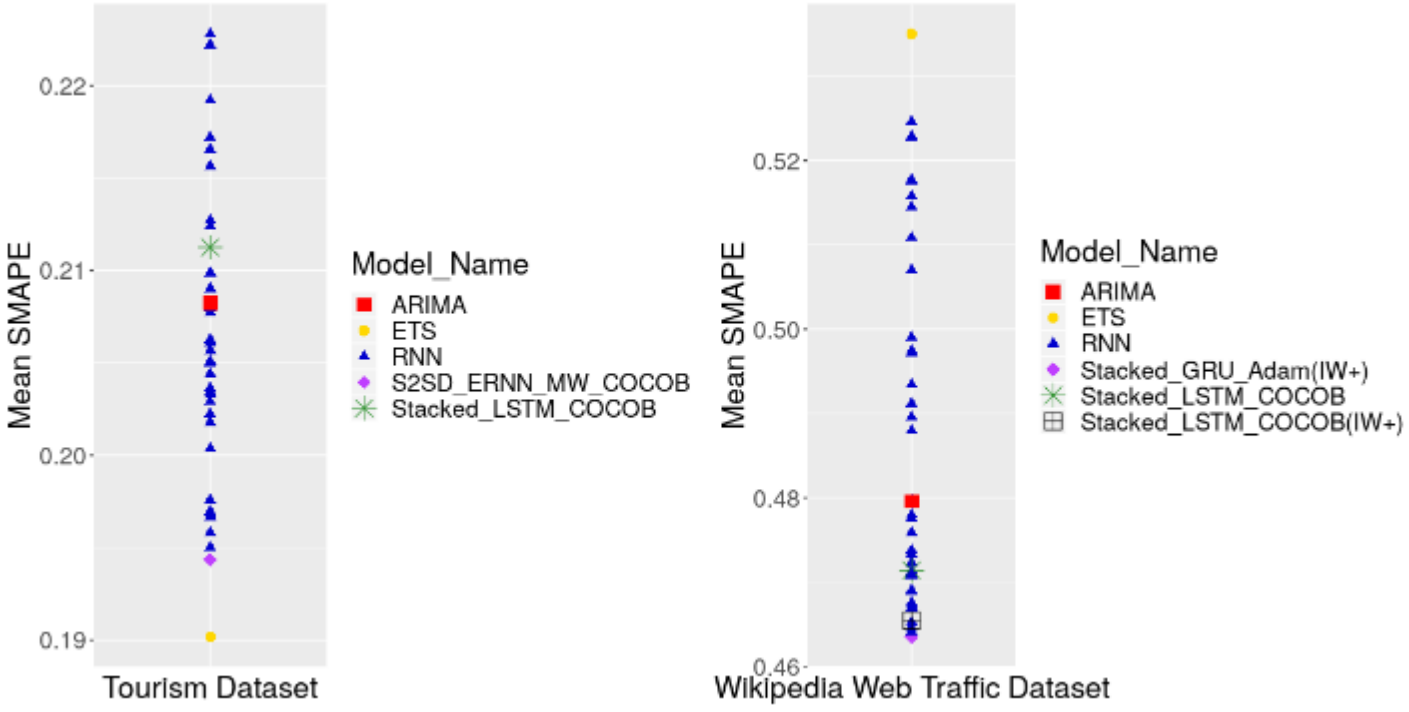
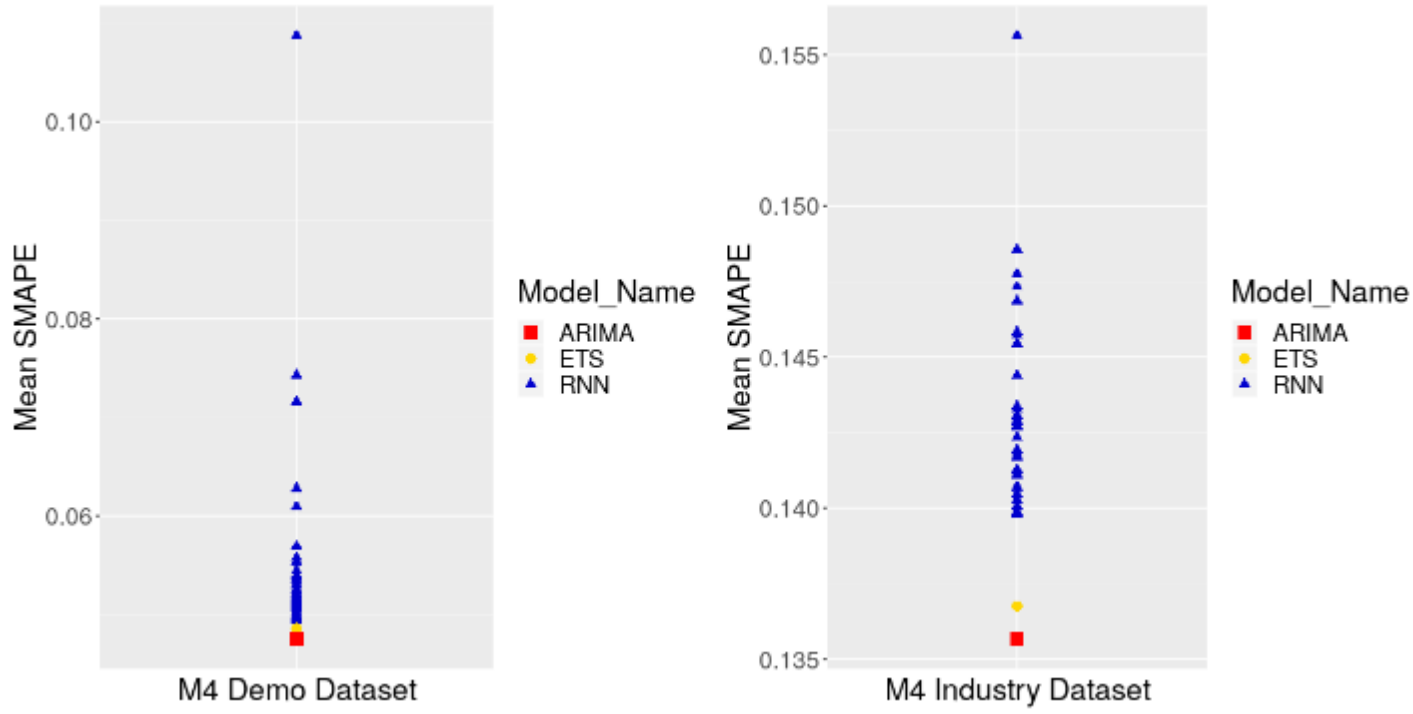


Figure 22: Performance of RNNs Compared to Traditional Univariate Techniques in Different M4 Categories (1/3)



- M3 and M4 span various domains, including microeconomics, macroeconomics, industry, finance, demography, and other sectors.

Figure 22: Performance of RNNs Compared to Traditional Univariate Techniques in Different M4 Categories (2/3)

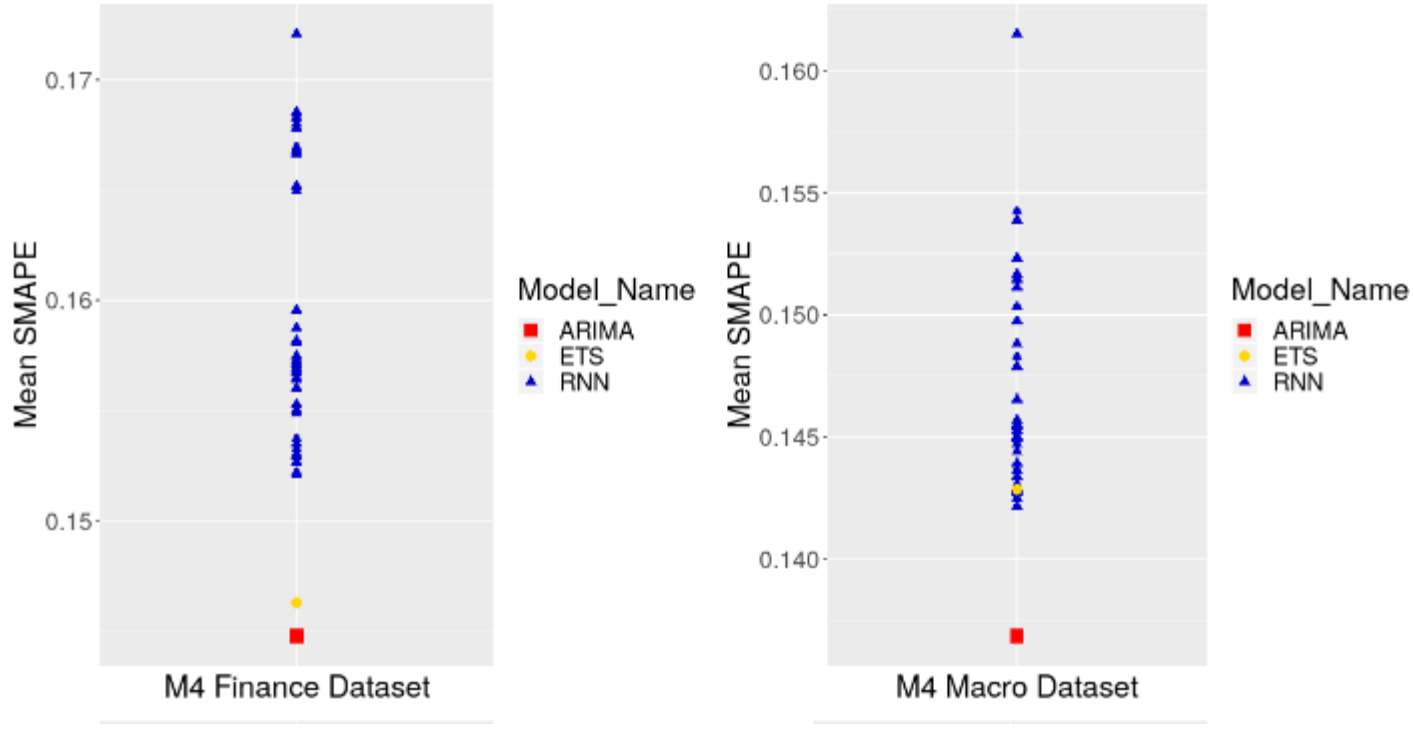
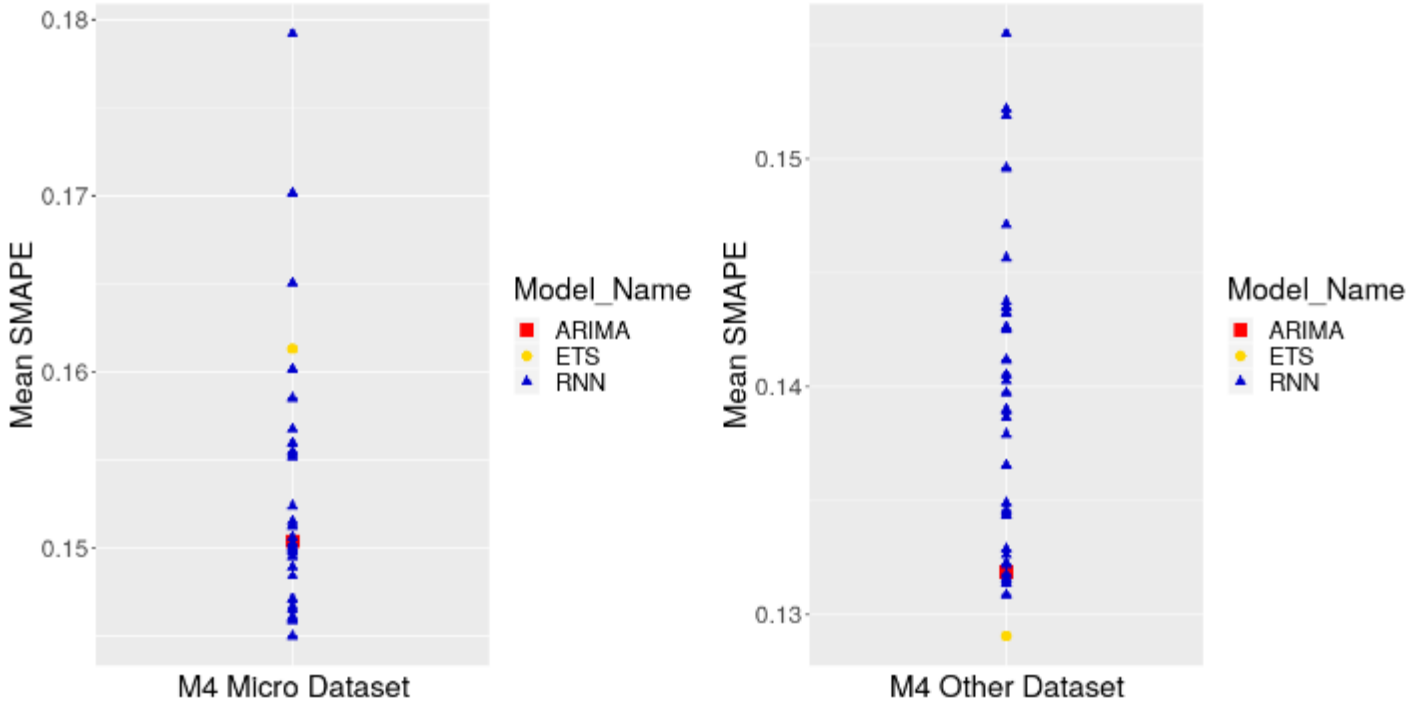


Figure 22: Performance of RNNs Compared to Traditional Univariate Techniques in Different M4 Categories (3/3)



Recap of Datasets Used

- CIF include both artificially generated series and series originating from the banking domain.
- M3 and M4 span various domains, including microeconomics, macroeconomics, industry, finance, demography, and other sectors.
- The NN5 (daily cash withdrawal amounts from UK ATMs), Wikipedia web traffic, and the Tourism datasets contain non-negative series, meaning that they also have 0 values.

Discussion

Comment. I haven't examined them in details, but I suspect the issue stems from inappropriate benchmarking and/or mishandling of time series characteristics like non-normality and seasonality.

For example,

- Are ETS and ARIMA suitable benchmarks for these datasets?
- Can we trust RNNs to perform reliably without explicit assumptions?

Discussion

The paper was published in 2021. Two of its authors later published "Forecast evaluation for data scientists: common pitfalls and best practices" in 2023, highlighting:

1. ML and DL techniques (including transformer-based architectures) are becoming competitive in time series forecasting.
2. ML researchers often use inadequate benchmarking (e.g., ignore non-stationary or non-normality) or flawed evaluation methods, leading to misleading claims of superiority.
3. Best practices are outlined with respect to the different steps such as data partitioning, error metrics, statistical testing, and more.

Thank you!

Thoughts?

References

Hewamalage, Hansika, Christoph Bergmeir, and Kasun Bandara. 2021. "Recurrent Neural Networks for Time Series Forecasting: Current Status and Future Directions." Interna- tional Journal of Forecasting 37 (1): 388–427.

Hewamalage, Hansika, Klaus Ackermann, and Christoph Bergmeir. 2023. "Forecast Eval- uation for Data Scientists: Common Pitfalls and Best Practices." Data Mining and Knowledge Discovery 37 (2): 788–832.

Olah, Christopher. 2015. "Understanding LSTM Networks." https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Supplement

Hyperparameter Tuning

The experiments require the proper tuning of a number of hyperparameters associated with the model training.

Key hyperparameters important for training include:

1. Batch size
2. Number of Epochs
3. Learning Rate
4. Cell Dimension
5. Number of Hidden Layers
6. Etc.

Hyperparameter Tuning - Details

1. Batch size

- the number of training samples or sequences processed simultaneously by the network in one forward and backward pass before updating its parameters.

2. Number of Epochs

- the number of full forward and backward passes across the full dataset are required for the optimal training.

3. Learning Rate

- controls how much the network's parameters are adjusted during training in response to the gradients of the loss function.

Hyperparameter Tuning - Details

4. Cell Dimension

- the size of the hidden state vector, which corresponds to the number of neurons inside each recurrent cell.

5. Number of Hidden Layers

- determines how many recurrent layers are layered on top of each other.

6. Etc.

Hyperparameter Tuning - Cont.

For tuning hyperparameters, there are several techniques, including:

1. Manual Search, also known as Manual Hyperparameter Tuning
2. Grid Search
3. Random Search

Table 4: Initial Hyperparameter Ranges

Dataset	Batch Size	Epochs	Epoch Size	Std. Noise	L2 Reg.	Cell Dim.	Layers	Std. Initializer	Learning Rate	
									Adam	Adagrad
CIF (12)	10 - 30	3 - 25	5 - 20	0.01 - 0.08	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
CIF (6)	2 - 5	3 - 30	5 - 15	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 5	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
NN5	5 - 15	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Mic)	40 - 100	3 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Mac)	30 - 70	3 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Ind)	30 - 70	3 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Dem)	20 - 60	3 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Fin)	20 - 60	3 - 30	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M3(Oth)	10 - 30	3 - 30	5 - 20	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Mic)	1000 - 1500	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Mac)	1000 - 1500	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Ind)	1000 - 1500	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Dem)	850 - 1000	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Fin)	1000 - 1500	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 50	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
M4(Oth)	50 - 60	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
Wikipedia	200 - 700	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9
Tourism	10 - 90	3 - 25	2 - 10	0.0001 - 0.0008	0.0001 - 0.0008	20 - 25	1 - 2	0.0001 - 0.0008	0.001 - 0.1	0.01 - 0.9

Table 4: Initial Hyperparameter Ranges

Figure 5: Stacked Architecture

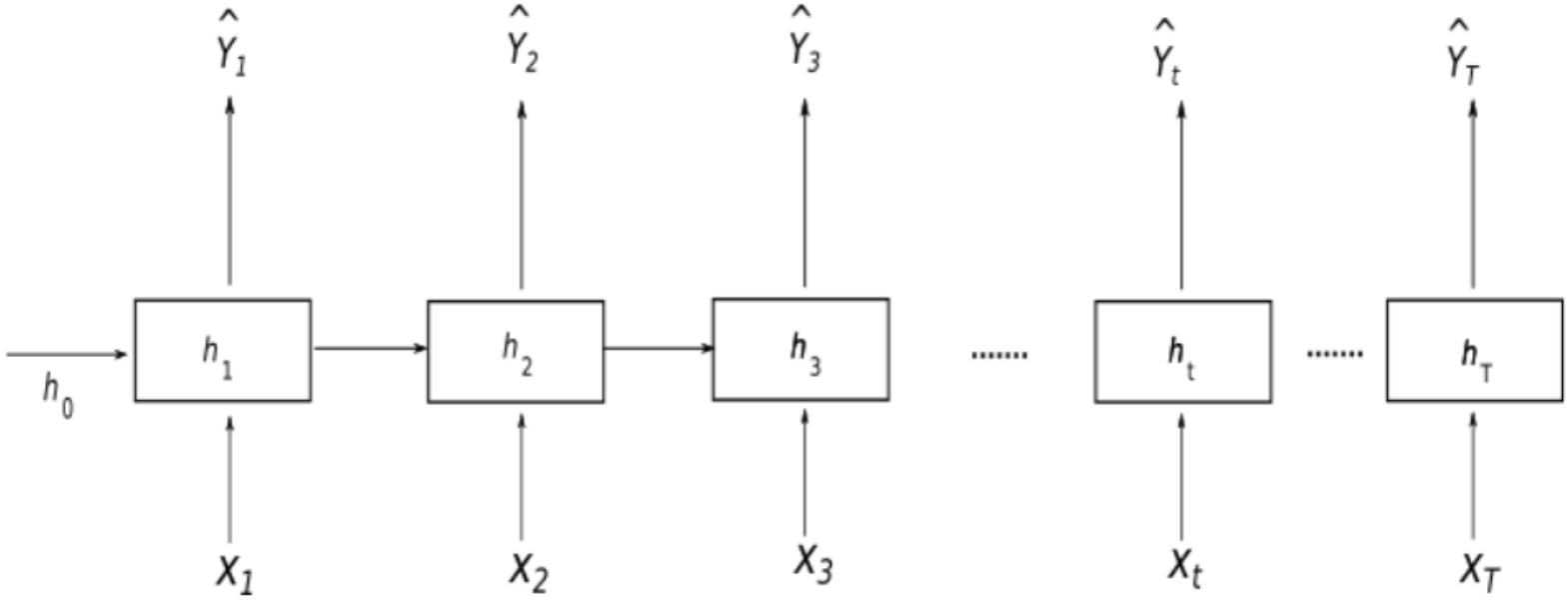


Figure 5: Stacked Architecture

Figure 6: Multi-layer Stacked Architecture

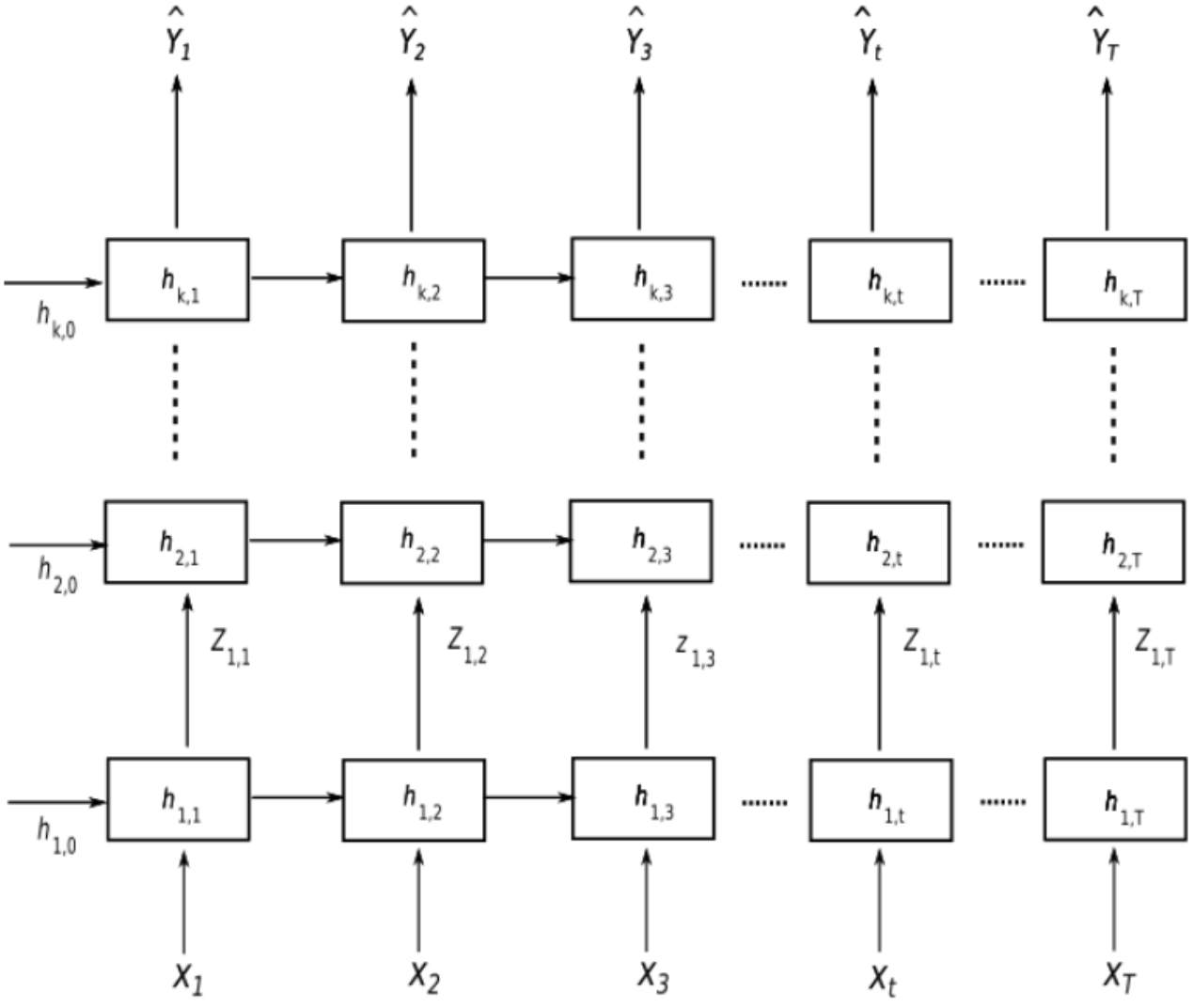


Figure 6: Multi-layer Stacked Architecture

Table 5: Mean SMAPE Results (1/6)

Model Name	CIF	Kaggle	M3	NN5	Tourism	M4
auto.arima	11.7	47.96	14.25	25.91	19.74	**13.08**
ets	11.88	53.5	14.14	21.57	**19.02**	13.53
Pooled Regression Lags 10	*14.67	*122.43†	*14.73†	*30.21†	*31.29†	*14.18†
Pooled Regression Lags 10 Bayesian Regularized	*14.67	*122.43†	*14.85†	*30.21†	*31.29†	*14.18†
Pooled Regression Lags 10 Regularized	*14.52	*123.94†	*14.75†	*30.21†	*31.29†	*14.19†
Pooled Regression Lags Window	*12.89	**47.47	*14.36†	*22.41	*21.1†	*13.75†
Pooled Regression Lags Window Bayesian Regularized	*12.89	**47.55	*14.42†	*22.41	*21.1†	*13.75†
Pooled Regression Lags Window Regularized	*12.89	**47.46	*14.38†	*22.41	*21.1†	*13.75†
Unpooled Regression Lags 10	15.35	*75.39†	14.81†	*30.06†	*27.39†	*13.38†
Unpooled Regression Lags 10 Bayesian Regularized	12.85	*84.36†	*15.99†	*30.21†	*30.41†	*14.2†
Unpooled Regression Lags 10 Regularized	*12.74	*94.28†	15.32†	*30.06†	*27.61†	*14†
Unpooled Regression Lags Window	14.34	*56.75†	14.93	*22.63	*20.71†	*13.46†
Unpooled Regression Lags Window Bayesian Regularized	12.46	*60.66†	*15.72†	*23.15	*21.58†	*14.15†
Unpooled Regression Lags Window Regularized	11.61	54.61†	14.72	*22.67	*21.13†	*13.65†
S2S GRU adagrad	11.57	51.77†	*15.91†	28.33	*20.57†	*13.53†
S2S GRU adam	11.66	49.9†	14.53†	*28.61	*20.62†	*14.52†
S2S GRU cocob	10.79	**46.89	*14.74†	28.36	*20.35†	*13.61†
S2S LSTM adagrad	11.23	51.77†	14.29†	28.34	*20.9†	*14.13†
S2S LSTM adam	10.71	**49.34‡	14.77†	*28.04	*20.5†	*13.99†
S2S LSTM cocob	11.04	52.46†	14.6	24.77	*20.51†	*14.02†
S2S ERNN adagrad	10.47	51.45†	*14.7†	*28.99	*20.44†	*14.24†
S2S ERNN adam	11.19	50.7†	*14.66†	*28.32	*20.63†	*13.86†
S2S ERNN cocob	10.48	**48.95†	*14.56†	*26.35	*20.77†	*13.8†

Table 5: Mean SMAPE Results (2/6)

S2SD GRU MW adagrad	11.6	51.77†	**14.11**	28.73	19.7	*13.38†
S2SD GRU MW adam	10.66	51.76†	14.78†	*25.17	19.5	*13.79†
S2SD GRU MW cocob	10.3	48.8†	14.38†	*27.92	19.67	*13.57†
S2SD GRU NMW adagrad	10.65	*52.29†	*14.27†	28.73	*20.99†	*13.51†
S2SD GRU NMW adam	10.41	**46.9	14.35†	*25.42	*20.5†	*13.92†
S2SD GRU NMW cocob	10.48	**47.33	14.35†	24.8	*20.8†	*13.75†
S2SD LSTM MW adagrad	11.58	51.76†	14.19	28.73	*20.36†	*13.39†
S2SD LSTM MW adam	10.28	51.09†	14.68	*26	*20.18	*13.76†
S2SD LSTM MW cocob	11.23	51.77†	14.45	*24.29	19.76	*13.59†
S2SD LSTM NMW adagrad	10.81	*52.28†	*14.39†	28.73	*20.22†	*13.5†
S2SD LSTM NMW adam	10.34	**47.37	14.43†	25.36	*20.33†	*13.62†
S2SD LSTM NMW cocob	**10.23**	49.74†	14.31†	*27.82	*20.29†	*13.66†
S2SD ERNN MW adagrad	11.34	51.58†	14.2	28.58	19.58	*13.42†

Table 5: Mean SMAPE Results (3/6)

S2SD ERNN MW adam	10.73	49.12†	14.75†	*27.49	19.69	*13.83†
S2SD ERNN MW cocob	10.39	**47.96†	14.73†	*25.58	19.44	*14.01†
S2SD ERNN NMW adagrad	10.31	49.72†	*14.86†	28.65	*21.27†	*13.52†
S2SD ERNN NMW adam	10.3	**47.76†	*14.79†	*29.1	20.04	*14.32†
S2SD ERNN NMW cocob	10.28	**47.94†	*14.94†	*26.37	19.68	*13.57†
Stacked GRU adagrad	10.54	**47.23	14.64	21.84‡	*21.56†	*14.15†
Stacked GRU adam	10.55	**47.11	14.76	21.93‡	*21.92†	*14.29†
Stacked GRU cocob	10.54	**46.73	14.67	22.18‡	*21.66†	*14.57†
Stacked LSTM adagrad	10.51	**47.12	14.34	22.12‡	*20.61†	*13.97†
Stacked LSTM adam	10.51	**47.13	14.39	**21.53**‡	*22.28†	*14.23†
Stacked LSTM cocob	10.4	**47.14	14.44	21.61‡	*21.12†	*14.13†
Stacked ERNN adagrad	10.53	**47.79	14.73	23.83‡	*21.24	*14.3†
Stacked ERNN adam	10.51	**46.7	14.71	23.53‡	*22.22†	*14.51†
Stacked ERNN cocob	10.57	**47.38	14.93	23.59‡	*21.72†	*14.24†
Stacked GRU adagrad(IW+)	-	**47.09	-	23.43	-	-
Stacked GRU adam(IW+)	-	**46.35	-	24.96	-	-
Stacked GRU cocob(IW+)	-	**46.46	-	21.92‡	-	-
Stacked LSTM adagrad(IW+)	-	**46.41	-	22.97	-	-
Stacked LSTM adam(IW+)	-	**46.52	-	21.59‡	-	-
Stacked LSTM cocob(IW+)	-	**46.54	-	21.54‡	-	-
Stacked ERNN adagrad(IW+)	-	**47.08	-	22.96	-	-
Stacked ERNN adam(IW+)	-	**46.76	-	22.5‡	-	-
Stacked ERNN cocob(IW+)	-	**47.59	-	23.13	-	-

Table 5: Mean SMAPE Results (4/6)

NSTL S2S GRU adagrad	*20.02†	*68.6†	*18.32†	*30.98†	*199.97†	-
NSTL S2S GRU adam	*16.98†	50.48†	*18.52†	*28.09†	*48.2†	-
NSTL S2S GRU cocob	*17.96†	50.44†	*17.56†	*26.01	*95.19†	-
NSTL S2S LSTM adagrad	*17.54†	*71.25†	*17.75†	*29.52†	*41.41†	-
NSTL S2S LSTM adam	*16.11†	**47.8†	*17.37†	*36.33†	*32.69†	-
NSTL S2S LSTM cocob	*17.8†	*68.36†	*18.42†	*30.29†	*48.44†	-
NSTL S2S ERNN adagrad	*19.98†	51.4†	*19.73†	*26.33†	*49.83†	-
NSTL S2S ERNN adam	*18.12†	**49.84†	*19.06†	*29.81†	*51.24†	-
NSTL S2S ERNN cocob	*17.97†	51.69†	*18.85†	*31.77†	*43.68†	-
NSTL S2SD GRU MW adagrad	*16.12†	51.08†	*20.87†	*24.45	*199.97†	*19.47†
NSTL S2SD GRU MW adam	*13.74	51.2†	*18.09†	*24.08	*36.02†	-
NSTL S2SD GRU MW cocob	*17.77†	51.71†	*19.48†	*24.31	*37.91†	-
NSTL S2SD GRU NMW adagrad	*14.42	**46.57	*15.8†	*25.43	*199.97†	-
NSTL S2SD GRU NMW adam	*15.71†	**46.08	*16.17†	*23.85	*38.63†	-
NSTL S2SD GRU NMW cocob	*15.05†	**47.69†	*16.14†	*24.85	*31.57†	-
NSTL S2SD LSTM MW adagrad	*27.06†	51.24†	*21.29†	*40.96†	*40.27†	-
NSTL S2SD LSTM MW adam	*15.23	51.11†	*19.33†	*25.7	*38.52†	-
NSTL S2SD LSTM MW cocob	*18.61†	49.83†	*23.05†	*24.14	*24.34†	-
NSTL S2SD LSTM NMW adagrad	*14.81†	49.77†	*15.9†	*25.03	*25.47†	-
NSTL S2SD LSTM NMW adam	*14.84	**46.07	*17.05†	*24.55	*23.91†	-
NSTL S2SD LSTM NMW cocob	*22.74†	**47.72†	*16.35†	*25.31	*31.52†	-
NSTL S2SD ERNN MW adagrad	*19.19†	50.71†	*19.22†	*27.35†	*22.29†	-
NSTL S2SD ERNN MW adam	*13.92	49.69†	*18.55†	*24.52	*30.79†	-
NSTL S2SD ERNN MW cocob	*14.88†	50.82†	*16.42†	*24.87	*26.16†	-
NSTL S2SD ERNN NMW adagrad	*15.78†	**47.41	*17.22†	*27.47†	*38.19†	-
NSTL S2SD ERNN NMW adam	*15.02	**46.37	*16.34†	*22.81	*28.55†	-
NSTL S2SD ERNN NMW cocob	*14.24	48.27†	*15.26†	*23.22	*25.82†	-

Table 5: Mean SMAPE Results (5/6)

NSTL Stacked GRU adagrad	*16.77†	**45.68‡	*16.39†	*24.39	*199.97†	-
NSTL Stacked GRU adam	*16.34†	****45.62**‡**	*16.76†	*23.47	*38.66†	-
NSTL Stacked GRU cocob	*17.64†	**46.35	*16.23†	*23.55	*41.52†	-
NSTL Stacked LSTM adagrad	*17.08†	**45.88‡	*16.69†	*25.1	*23.46†	-
NSTL Stacked LSTM adam	*15.83†	**46.12	*17.54†	*24.44	*21.73†	-
NSTL Stacked LSTM cocob	*15.27†	**45.9	*17.97†	*26.28†	*25.24†	*17.71†
NSTL Stacked ERNN adagrad	*13.25	**46	*16.78†	*24.22	*25.36†	-
NSTL Stacked ERNN adam	*16.05†	**45.77‡	*16.43†	*23.04	*20.67	-
NSTL Stacked ERNN cocob	*15.74†	**46.02	*17.32†	*23.55	*28.71†	-
NSTL Stacked GRU adagrad(IW+)	-	**45.9	-	*23.53	-	-
NSTL Stacked GRU adam(IW+)	-	**45.69‡	-	22.2	-	-
NSTL Stacked GRU cocob(IW+)	-	**46.01	-	*22.78	-	-
NSTL Stacked LSTM adagrad(IW+)	-	**45.68‡	-	*23.15	-	-

Table 5: Mean SMAPE Results (6/6)

NSTL Stacked LSTM adam(IW+)	-	**46.32‡	-	22.45	-	-
NSTL Stacked LSTM cocob(IW+)	-	**46.12	-	22.66	-	-
NSTL Stacked ERNN adagrad(IW+)	-	**45.69‡	-	*22.79	-	-
NSTL Stacked ERNN adam(IW+)	-	**45.92‡	-	22.13	-	-
NSTL Stacked ERNN cocob(IW+)	-	**45.79‡	-	*22.8	-	-

Table 5: Mean SMAPE Results