Recurrent neural network implementations on the M4 dataset

Thomas J. Delaney

Contents

1 Introduction

The purpose of this document is to describe the recurrent neural networks (RNNs) applied to the M4 dataset as an attempt to forecast the financial series in the M4 dataset.

2 Objectives

Our main objective in this experiment was to create a forecasting method similar to that of Smyl et al (2018), which won the M4 forecasting competition [3]. More specifically, we wanted to create forecasting method that blended statistical forecasting with machine learning methods using recurrent neural networks. So, our aims were:

- To train non-seasonal and seasonal forecasting methods using statistical techniques and RNNs together.
- To use these methods to forecast the financial series in the M4 dataset.
- To compare the performance of these methods to simpler statistical methods.
- To compare the performance of these methods to the M4 winning statistical-RNN hybrid model.

Our hope (rather than expectation) was that a model simpler than that in [3] could be used to forecast the financial time series.

3 Background

For background on the M4 competition, see [1] or [2].

The winner of the M4 competition was a hybrid forecasting model that combined simple exponential smoothing methods with recurrent neural networks. It was developed by Slawek Smyl while working at Uber. In this model, simple exponential smoothing is used to estimate a multiplicative level and seasonality of each time series. These level and seasonality components are used to scale the time series before the time series is passed to the RNN. An RNN with a different architecture is used for each time series frequency. Each RNN uses LSTM cells and dilation, and the attention mechanism is used for some frequencies.

The learning mechanism and model is hierarchical in nature. The exponential smoothing parameters are learned for each individual time series, whereas the weights in the RNN are learned across all the scaled time series. For more information on this model see [3] or [4]. For more information on recurrent neural networks, see [4].

The statistical modelling in this experiment was performed in `R` using the `forecasting` package. We specifically used an *ets model* for modelling both seasonal and non-seasonal data. For more information about ets models, other statistical models, and the `forecasting` package see [5].

The RNNs in this experiment we created and trained in `R` using *RStudio* and *Keras*. Keras is a high-level neural networks API enabling quick and easy building of and experimentation with neural networks. We used *tensorflow* as the back-end for Keras. For for information on Keras, see [6]. For more information on Tensorflow, see[7].

4 Methods

4.1 Statistical Modelling

In this section, we describe the statistical models applied to the time series. Bare in mind that these models were not intended to accurately model or forecast the time series. The intention was to use the level and seasonal components extracted to scale the time series before passing the scaled series into the RNN.

4.1.1 Non-seasonal

To statistically model the yearly time series, we used a non-seasonal ETS model with a multiplicative error and no trend component. This models the time series as a level component with multiplicative white noise error. The model is defined by

$$y_t = \ell_{t-1}(1 + \epsilon_t) \tag{1}$$
$$\ell_t = \ell_{t-1}(1 + \alpha\epsilon_t) \tag{2}$$

where y_t is the time series, ℓ_t is the level component, and α is a smoothing parameter.

4.1.2 Seasonal

To statistically model quarterly time series, we used an ETS model with a multiplicative seasonality, multiplicative error, and no trend component. This models the time series as a level component with a multiplicative seasonal component, and a multiplicative white noise error. The model is defined by

$$y_t = \ell_{t-1}s_{t-m}(1 + \epsilon_t) \tag{3}$$
$$\ell_t = \ell_{t-1}(1 + \alpha\epsilon_t) \tag{4}$$
$$s_t = s_{t-m}(1 + \gamma\epsilon_t) \tag{5}$$

where y_t is the time series, ℓ_t is the level component, α is a level smoothing parameter, s_t is the seasonal component, γ is the seasonal smoothing parameter, and m is the number of time points (or seasons) in a full seasonal cycle (e.g. 4 for quarterly data).

4.2 Recurrent Neural Network

The RNN that we used in this experiment is fairly simple compared to those used in [3]. Time constraints were the main reason for this. The network consisted of an input layer, two hidden layers, and an output layer.

The first hidden layer consisted of 32 gated recurrent units (GRUs), with $\tanh$ output activation and *hard sigmoid* recurrent activation (see 4.2.1). A standard dropout rate of 10% and recurrent dropout rate of 50% were implemented.

The second hidden layer consisted of 64 GRUs. This time a rectified linear unit output activation was used.The same dropout rates were used.

The output layer consisted of a number of densely connected units equal to the number of points to be forecast (6 for yearly, 8 for quarterly).

Note that we used only the last 30 elements of each time series in our model. This was to keep the training time reasonable short. We also felt that given the forecast horizon for the non-seasonal data was 6 time steps, and the forecast horizon for seasonal data was 8 time steps, 30 time steps should be enough to train a model for forecasting.

4.2.1 Hard sigmoid activation function

The hard sigmoid function is a piece-wise linear approximation of the sigmoid function. When plotted in two dimensions, the function consists of three linear segments, one along the x-axis, one with a slope of $\frac{1}{5}$ and intercept of $\frac{1}{2}$, and one parallel to the x-axis with $f(x) = 1$. The function can be defined by

$$f(x) = \begin{cases} 0, & \text{if } x < -\frac{5}{2} \\ \frac{x}{5} + \frac{1}{2}, & \text{if } -\frac{5}{2} \leq x \leq \frac{5}{2} \\ 1, & \text{if } x > \frac{5}{2} \end{cases} \tag{6}$$

Figure 1 shows the standard sigmoid and the hard sigmoid over the domain $[-10, 10]$ for comparison [1].

This function is commonly used in place of the sigmoid function because it's much quicker to calculate the hard sigmoid. Specifically, the calculations for e^{-x} are no longer necessary, and only linear operations need to be performed.

4.3 Combining the statistical model and RNN

4.3.1 Non-seasonal

In order to combine the statistical model with the recurrent neural network, and attempt to use the power of both we did the following:

1. We split the dataset of non-seasonal M4 series into training, validation, and test sets.

2. We fit the non-seasonal ETS model described in section 4.1.1 to each of the training series. This returned a level series, and a residual series for each time step in each series.

3. We trained the RNN to forecast the residual series, using the mean absolute error as the loss function. If we examine equation 1, we can see that forecasting the residuals is equivalent to forecasting a scaled and translated version of the time series, namely $\frac{y_t}{\ell_{t-1}} - 1$.

4. In order to obtain a descaled and detranslated forecast, we used the values for ϵ_t given to us by the RNN output along with equations 1 and 2 to calculate y_t and ℓ_t for each timestep in the forecast horizon.

We used the validation set during the training of the RNN, and to quantify the performance of our hybrid method compared to the baseline method.

[1] See the code of the keras implementation of this function in Python here: `https://github.com/keras-team/keras/blob/master/keras/backend/tensorflow_backend.py#L3394`

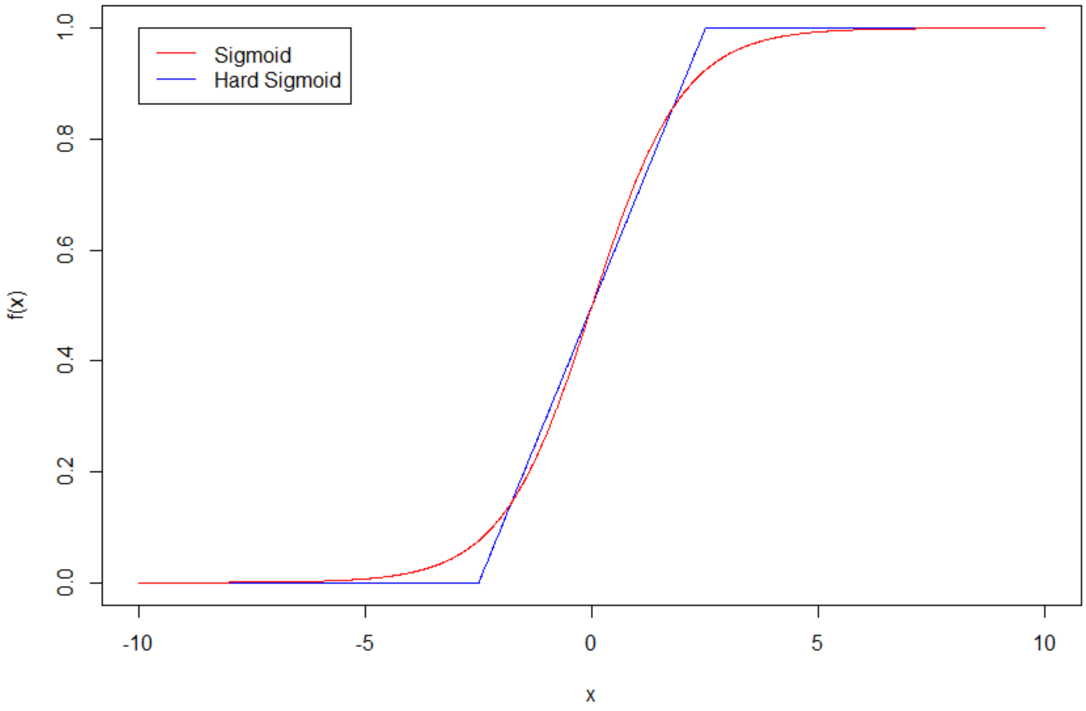

Figure 1: The sigmoid and hard sigmoid functions. The hard sigmoid is a linear approximation of the sigmoid that is commonly used as the recurrent activation in GRU or LSTM networks.

4.3.2 Seasonal

The process for the seasonal time series was similar to that for the non-seasonal, except that the fitted ets model (as described in section 4.1.2) returned a seasonal series as well as a level series and a residual series. The residual series was equivalent to $\frac{y_t}{\ell_{t-1}s_{t-1}} - 1$ In order to obtain the final forecast, we had to use the level and seasonal series with equations 3, 4, and 5 to descale and deseasonalise the output of the RNN.

4.4 Baseline Methods

In order to assess the performance of our hybrid model, we compared its forecasting performance to that of a simpler statistical model. We chose different simple models for non-seasonal and seasonal data.

4.4.1 Non-seasonal

For non-seasonal forecasting we used a random walk model with a drift component as the baseline model.

4.4.2 Seasonal

For seasonal forecasting, we used a seasonal naïve method as the baseline model.

5 Results

5.1 Non-Seasonal Model

5.1.1 RNN Training

There were 23,000 yearly time series in the M4 dataset. We split these into a training set of 16,000 series, a validation set of 3500, and a test set of 3500 series. We fit the model using the training and validation set. We used the mean absolute error as the loss function. See figure 2 for the training and validation progression during training. Fitting took about 19 hours.

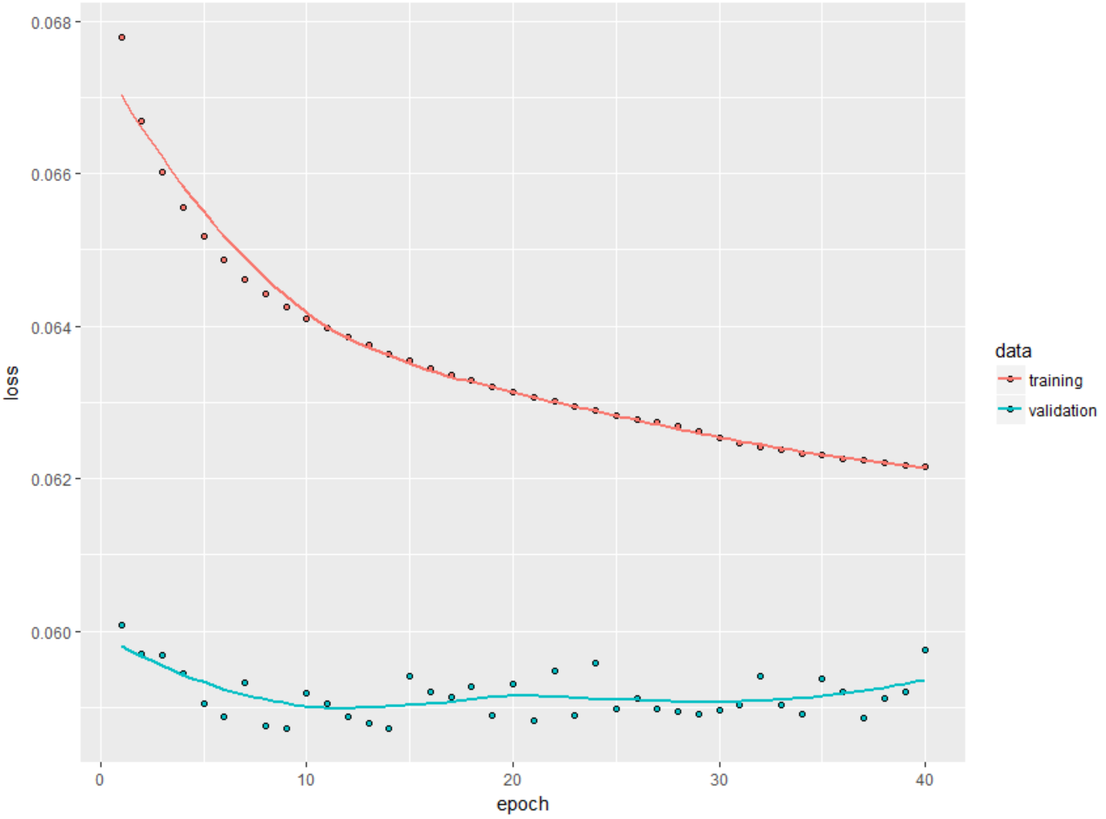

Figure 2: Training and validation loci during the fitting of the non-seasonal RNN model. Note that since we trained the RNN on the residuals of the ETS model, the values for the loss here are not in the same scale as the model's forecasts.

5.1.2 Comparison to baseline measure

As mentioned in section 4.4.1, the baseline statistical forecasting method chosen for the non-seasonal data was the random-walk with drift model (RW model). Using the MAE as the error measure, the RW model outperformed our hybrid model. The MAE of the hybrid model on the test set was 1183.167. The MAE of the RW model on the test set was 1017.714.

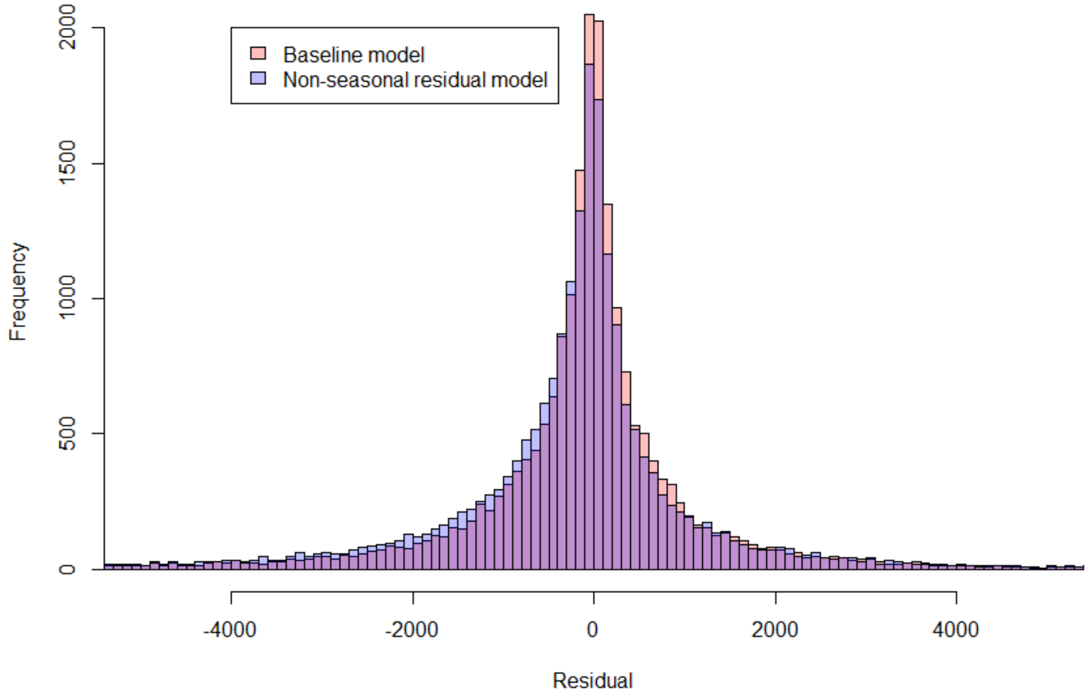

Figure 3: Distributions of the residuals of the non-seasonal hybrid model and the baseline RW model shown as overlapping histograms. Both sets of residuals are normally distributed, but the variance of the hybrid model's residuals is higher.

The distributions of the residuals for both models are shown in figure 3. Both sets of residuals are normally distributed, but the standard deviation of the RW model's residuals is 2273.058, and the standard deviation of the hybrid model's residuals is 2643.934.

5.1.3 Comparison to ETS model

Since the ETS model described in section 4.1.1 is used to scale the input to the RNN in our hybrid model, it makes sense to compare the performance of our model with that simpler ETS model. The ETS model outperformed our hybrid model just slightly. The MAE of the non-seasonal ETS model on the test set was 1159.775.

The distributions of the residuals for both models are shown in figure 4. Both sets of residuals are normally distributed, but the standard deviation of the ETS model's residuals is 2356.307, and the standard deviation of the hybrid model's residuals is 2643.934.

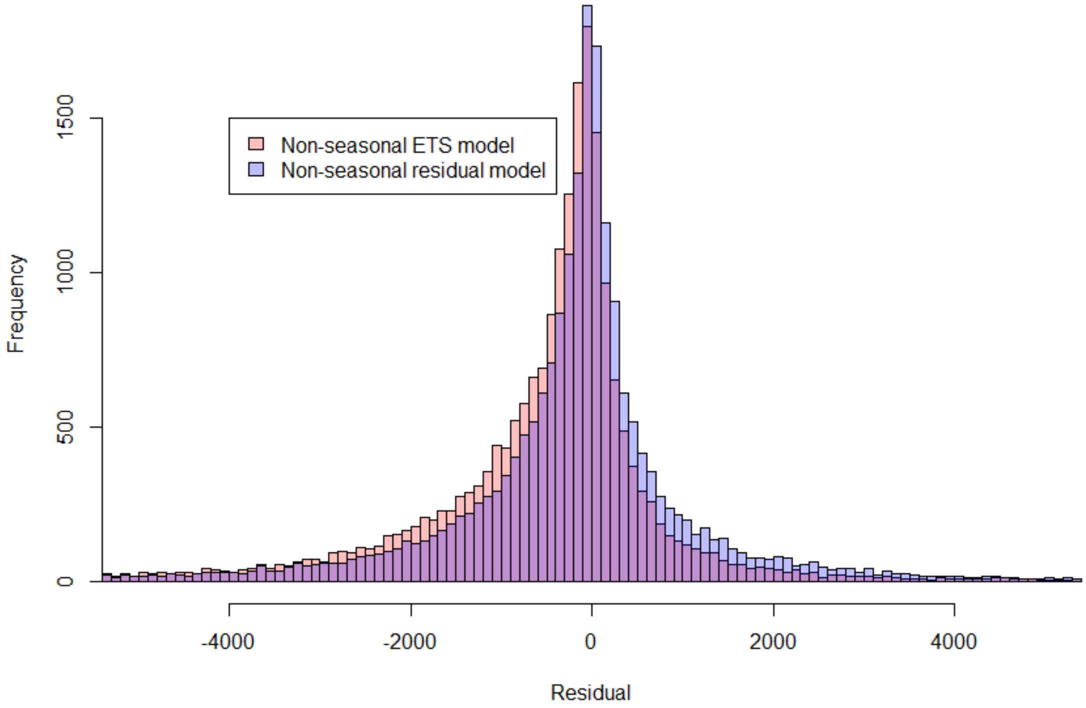

Figure 4: Distributions of the residuals of the non-seasonal hybrid model and the non-seasonal ETS model shown as overlapping histograms. Both sets of residuals are normally distributed, but the variance of the hybrid model's residuals is higher.

5.1.4 Comparison to Meta-learning model

5.2 Seasonal Model

5.2.1 RNN Training

There were $24{,}000$ quarterly time series in the M4 dataset. We divided these series into a training set of $17{,}000$ series, a validation set of 3500, and a test set of 3500 series. Similar to 5.1.1, we fit the model using the training and validation set. We used the mean absolute error as the loss function. See figure 2 for the training and validation progression during training. Fitting took about 19 hours. Since the validation loss is increasing while the training loss decreases, the model appears to be overfiitting throughout training.

5.2.2 Comparison to baseline measure

As mentioned in section 4.4.2 the baseline statistical model for seasonal data was the seasonal naïve model. Using the MAE, our hybrid model outperformed the baseline model. The MAE of the hybrid model on the test set was 718.4464. The MAE of the seasonal naïve model on the test set was 854.3212.

The distributions of the residuals for both models are shown in figure 6. Both sets of residuals are normally distributed, but the standard deviation of the seasonal naïve

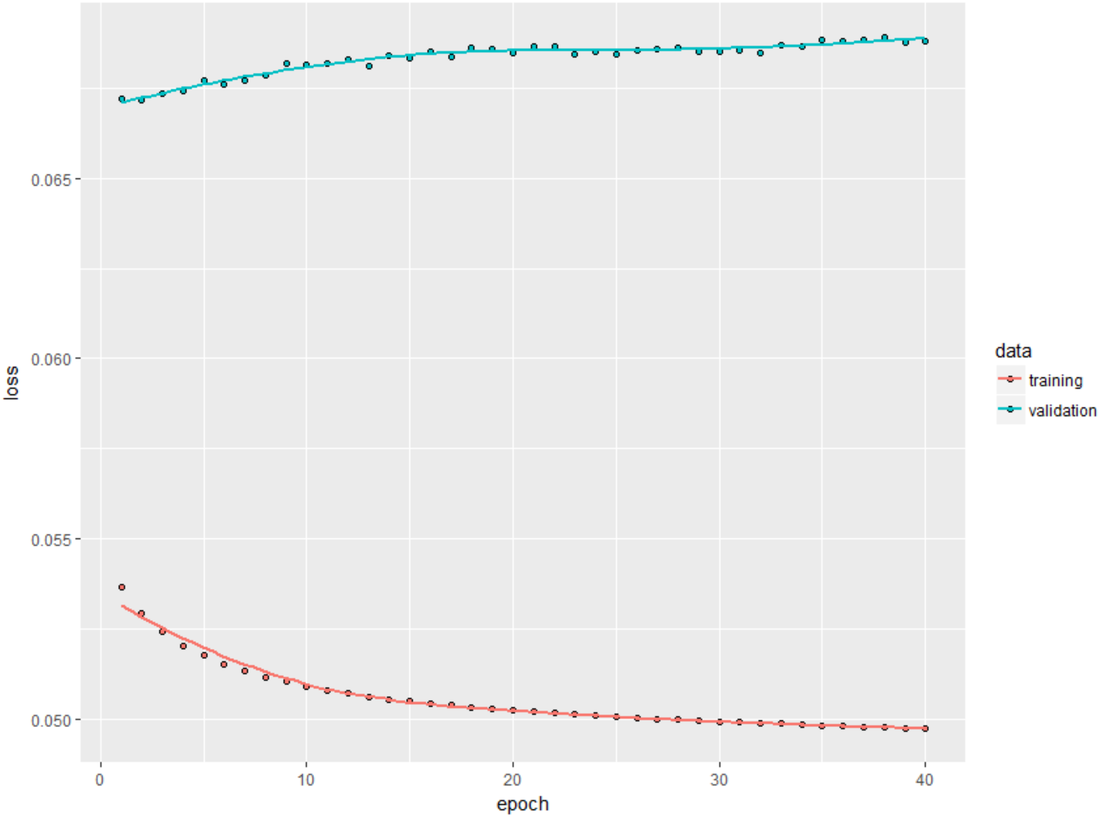

Figure 5: Training and validation loci during the fitting of the seasonal RNN model. Note that since we trained the RNN on the residuals of the ets model, the values for the loss here are not in the same scale as the model's forecasts.

model's residuals is 1781.011, and the standard deviation of the hybrid model's residuals is 1621.367.

5.2.3 Comparison to ETS model

The ETS model described in section 4.1.2 is used to the scale the seasonal data before using it as input for the RNN. So it makes sense to compare our hybrid model to the performance of that ETS model. The ETS model and our hybrid model had almost exactly the same MAE. The MAE of the ETS model was 718.6298.

The distributions of both models are shown in figure 7.

6 Conclusions

6.1 More Complexity Required

The hybrid model developed in Uber is much more complicated than the hybrid model described in this document. For example,

- Uber's networks us LSTM units as opposed to GRU units.

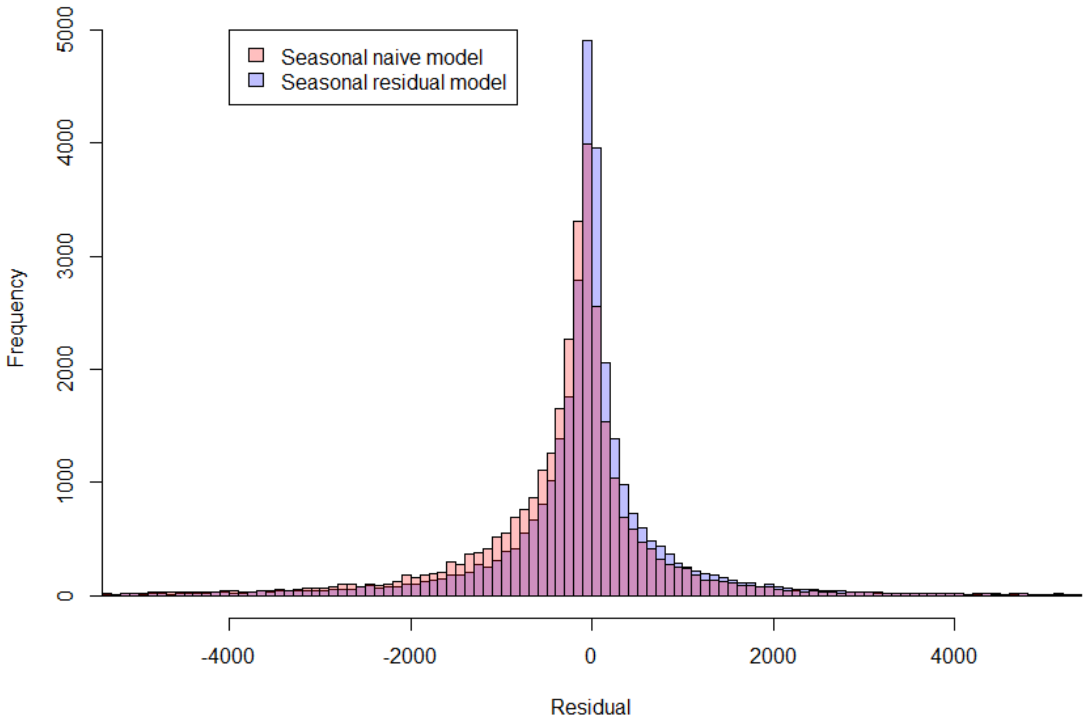

Figure 6: Distributions of the residuals of seasonal hybrid model and the baseline seasonal naïve model shown as overlapping histograms. Both sets of residuals are normally distributed, but the variance of the seasonal naïve model's residuals is higher.

- Uber's networks have more layers.
- The connections between the layers, and the time steps in Uber's networks are more complicated. Uber's networks make use of dilation and attention mechanisms.
- The process of scaling using the level and seasonal components is a part of Uber's learning process, as opposed to a pre-processing step, as it is here.

Given that our hybrid model was unable to outperform simple statistical methods, it is safe to conclude that additional complexity, like the complexity we see in Uber's model, is required for a more successful forecasting method.

6.2 Further Research

There is months of potential further work to be done. We could experiment with the training of the networks by perturbing the number of epochs, introducing early stopping, using different loss functions, using different optimisers etc.

We could experiment with the network itself by using LTSM units or JANET units in place of GRUs. We could add more layers. We could add attention mechanisms, or dilation mechanisms. We could add residual connections. We could add convolutional layers at shallower layers above the RNN.

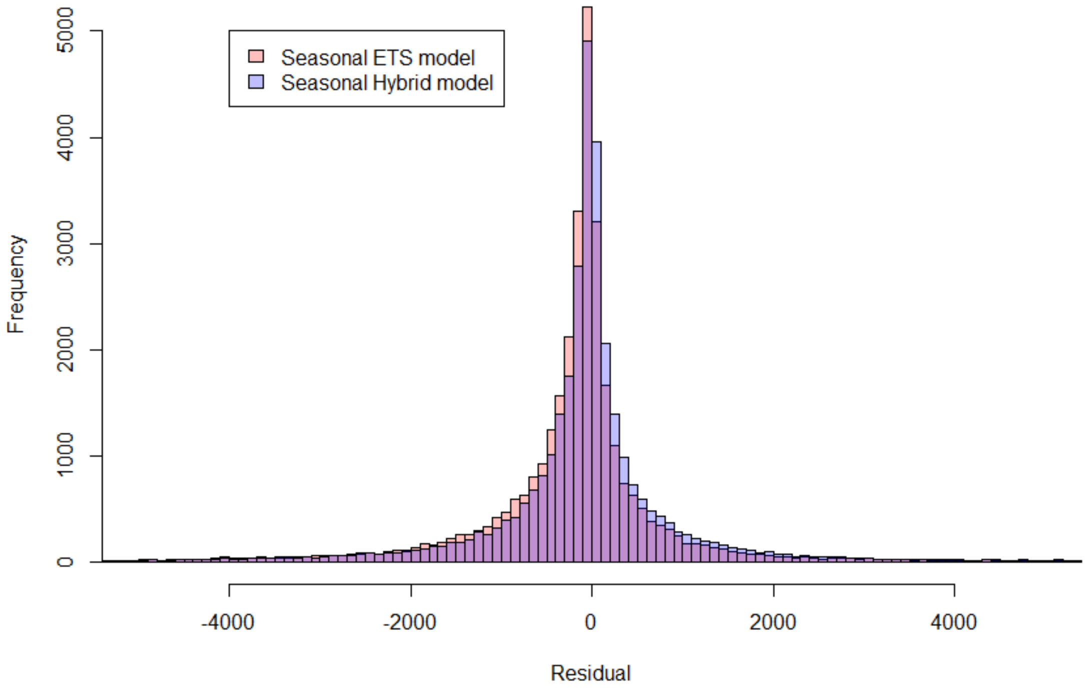

Figure 7: Distributions of the residuals of the seasonal hybrid model and the seasonal ETS model shown as overlapping histograms. The distributions are very similar.

Finally, we only trained and tested on the M4 dataset. The origins of these time series are somewhat foggy. Only the time points, values, and a 'type' classifier are given for each time series. It would be worthwhile using time series from more transparent alternative sources.

Dataset	Model	Mean Absolute Error
Yearly	Random Walk with Drift	1017.714
Yearly	Non-seasonal ETS	1159.775
Yearly	Non-seasonal Hybrid	1183.167
Quarterly	Seasonal Hybrid	718.446
Quarterly	Seasonal ETS	718.6298
Quarterly	Seasonal Naïve	854.3212

Table 1: A breakdown of the performances of the models applied to their respective datasets.

References

[1] Spyros Makridakis, *M4 Competition*. https://www.m4.unic.ac.cy/about/, University of Nicosia, (2018)

[2] Thomas Delaney, *Meta-learning for financial forecasting*. https://github.com/thomasjdelaney/Financial-Forecasting/blob/master/latex/meta-learning method review/meta-learning_method_review.pdf, (2018)

[3] Slawek Smyl, Jai Ranganathan, Andrea Pasqua, *M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model*. https://eng.uber.com/m4-forecasting-competition/, (2018)

[4] Thomas Delaney, *Forecasting Methods: An Overview*. https://github.com/thomasjdelaney/Financial-Forecasting/blob/master/latex/forecasting lit review/forecasting_lit_review.pdf, (2018)

[5] Rob J Hyndman, George Athanasopoulos, *Forecasting: Principles and Practice*. Monash University, Australia, (2018)

[6] Keras, *R interface to Keras*. https://keras.rstudio.com/, (2018)

[7] Tensorflow, *TensorFlow*. https://www.tensorflow.org/, (2018)

[8] Evangelos Spiliotis, Spyros Makridakis, Vassilios Assimakopoulos, *The M4 Competition in Progress*. https://www.m4.unic.ac.cy/wp-content/uploads/2018/06/Evangelos_Spiliotis_ISF2018.pdf, (2018)