Timo Meiendresch

Recurrent Neural Networks for Time Series Forecasting

Capstone Project
Machine Learning Engineer Nanodegree (Udacity)

Cologne 2019

Contents

Abstract

The year is 2019 A.D. Quantitative research is entirely occupied by Machine Learning (ML). Well, not entirely... One small group of indomitable forecasters still holds out against the invaders. And life is not easy for the ML evangelists who garrison the fortified camps of quantitative research.

1 Introduction

Time series forecasting is an essential tool in business, economics, and many other disciplines. The ability to accurately infer future variables given historic data is of huge value to decision makers.

In recent years, **Machine Learning (ML)** methods had huge success in many areas of quantitative research. Prominent examples include classification tasks, image processing, or text analysis. ML took many fields by storm and it seems that many quantitative discipline can benefit from giving in to the AI wave.

And yet, ML methods are rarely considered in time series forecasting. Research within this field indicated an inferior performance compared to traditional methods. Moreover, those methods are often easier to implement, are computationally cheap, and better interpretable. Important examples include the **Autoregressive Integrated Moving-Average (ARIMA)** and the various **Exponential Smoothing (ES)** methods.

Until recently, a widely shared notion was that complex methods for time series forecasting generally were not performing better than traditional ones (e.g. Hyndman, 2019). Among others, Makridakis et al. (2018) noted that there is only very limited scientific evidence which suggests that artificial neural networks may be a useful tool for time series forecasting.

Paradigm Shift

But, advances in recent years start to challenge this perception. For example, the winner of the most recent version of the M4 time series forecasting competition (Makridakis et al., 2019) was an elaborate hybrid of a **recurrent neural network (RNN)** architecture with ES methods (Smyl, 2019). Measured on a diverse dataset of 100,000 univariate time series from six domains and frequencies, this method outperformed traditional ones, pure statistical combinations as well as ML-based combination methods. The results of the competition showed the potential of recurrent networks for time series forecasting, leading to a surge of algorithms in this area. It also showed that

RNN could successfully be trained on highly diverse data. Commonly, RNN methods for time series foreasting are applied to a large set of homogeneous data.

Local vs. global methods

Moreover, the M4 competiton highlights an ongoing paradigm shift in the forecasting community. Traditional methods like ARIMA and ES are applied to individual series. These **local** methods estimate a number of parameters within a model space that is restricted by the respective structure of the model. Because these approaches rely on an explicit model structure they are referred to as **model-based** (Wang et al., 2019). Each time series is modeled and trained independently of other series in the data set. Hence, the individual model is independent of the total number of series in the as well as how similar these series are.

Also, with the emergence and availability of large sets of data, a new forecasting problem emerged. It is becoming increasingly necessary to implement large-scale forecasting systems that simultaneously predict many, often related, series. Examples include predictions of product sales in online retailers, household energy consumption, or web traffic. In contrast to local models, **global** methods enable the use of **cross-series learning**, i.e. during training process all available time series are used. This approach is supposed to yield a general representation of the entire data set. In theory, RNN-based methods should be able to learn across individual series and extract general patterns of the data as well as learn dependency relationships between the individual series which may improve forecasting accuracy. This may improve the forecast accuracy over local approaches.

There are various methods based on the idea of cross-series learning, in particular utilizing recurrent networks. The determinants of their performance is a relatively new question into which very little empirical research has so far been done.

In theory, RNN should be able to work as a universal function approximator if it is only powerful enough. In practice however, the learnability of RNNs is limited. A typical problem is that they do not converge during training. They are considered to be difficult to train. This is an important factor for practitioner's as training a powerful network on such a huge amount of data is computationally expensive.

For now, it seems that there are two key determinants that should be considered in the performance of the models:

Determinant 1. *Total number of individual time series in the underlying data set. How many series are in the dataset?*

Determinant 2. *Similarity of the individual time series in the data set. How diverse are the series?*

This leads to two hypothetically important caveats regarding the dataset which influence the performance. The total number of time series in the dataset as well as how diverse the series are with respect to each other.

Often, these methods lack rigorous comparison with previous methods and known data sets. This capstone project investigates the performance of three RNN-based methods developed in the last three years, namely:

- **Deep Autoregressive Recurrent Neural Networks (DeepAR)** - Wang et al. (2017)
- **Deep State Space Models (DeepState)** - Rangapuram et al. (2018)
- **Deep Factor Recurrent Neural Network Model (DF-RNN)** - Wange et al. (2019)

To evaluate these models I will apply them to the Hourly, Daily, Weekly and Monthly data of the M4 competition using the same measures of accuracy. Hence, a variety of benchmarks is available for comparison.

Note that these methods are designed to be applied to large sets of related data. In contrast, the M4 data set is a diverse dataset from many domains and frequencies.

To the best of my knowledge, no empirical investigation of these algorithms to the M4 benchmarks has been published yet. The primary contribution of this paper is to evaluate these algorithms on the well studied M4 data and compare the results to known benchmarks.

The rest of the paper is organized as follows. I first discuss ... in Section ....

2 Definition

2.1 Project Overview

This project compares three RNN-based algorithms for time series forecasting to the well studied M4 data and its benchmarks, including modern and traditional methods. The datasets include the original M4 data for hourly, daily, weekly, and monthly frequencies.

2.2 Problem Statement

Main goal of this paper is to evaluate the performance of three algorithms on the M4 data. The algorithms are DeepAR, DeepState, and DF-RNN. These results will be compared with traditionally widely used methods as well as state-of-the art methods that were part of the competition. The main goal is to assess their their competitiveness using default hyperparameters.

2.3 Metrics

The performance of the methods is evaluated using the same two measures that were used in the M4 competition. Forecast measures are still a highly active area of research and there is no consensus on which one is considered to be the best one. Hence, two widely used **scale-independent errors** were used in the competition. Scale-independent means they are independent of the unit of the time series. Because the competition evaluates accuracy across highly diverse time series, scale-independent measures are required. The two measures are:

- **Symmetric mean absolute percentage error (sMAPE)**
- **Mean absolute scaled error (MASE)**

The **sMAPE** belongs to the class of percentage errors which are unit-free. One drawback of the measure is that it is infinite/undefined if the target variable is zero for any t in the period of interest, and is highly skewed towards extreme values if the target variable is close to zero. The measure is defined as

$$sMAPE = \frac{1}{h}\sum_{t=1}^{h} \frac{2\,|Y_t - \hat{Y}_t|}{|Y_t| + |\hat{Y}_t|}$$

where Y_t is the post-sample value, $\hat{Y}_t$ the estimated forecast, and h the forecasting horizon.

Alternatively, Hyndman and Koehler (2006) propose the scaled error measure **MASE** when comparing forecast accuracy across series with different units. In this case the absolute error is scaled using the **Mean Absolute Error (MAE)** from a simple forecast method (either naive or seasonal naive method). The method is defined for seasonal data in the following way:

$$MASE = \frac{1}{h}\,\frac{\sum_{t=1}^{h}|Y_t - \hat{Y}_t|}{\frac{1}{n-m}\sum_{t=m+1}^{n}|Y_t - Y_{t-m}|}$$

where m indicates the seasonality of the series. In case of non-seasonal data ($m = 1$), $Y_t - Y_{t-m}$ simplifies to $Y_t - Y_{t-1}$ being the naive forecast error. The naive forecast predicts the next value to be equal to the last value $\hat{Y}_{t+1} = Y_t$. This yields the non-seasonal version of the MASE as

$$MASE = \frac{1}{h} \frac{\sum_{t=1}^{h} |Y_t - \hat{Y}_t|}{\frac{1}{n-m} \sum_{t=m+1}^{n} |Y_t - Y_{t-m}|}$$

In addition to measuring point forecast accuracy using MASE and sMAPE, prediction interval accuracy has also been introduced in the competition for the first time. For this, the **Mean Scaled Interval Score (MSIS)** has been used as accuracy measure. Furthermore, other widely used measures for evaluating forecast accuracy include the normalized sum of quantile losses for quantiles $p \in 0.5, 0.9$. These were the preferred measures in some of the mentioned papers which makes it difficult to use the empirical results of the paper directly. I decided to include them together with the MSIS only for reference and will not discuss these results explicitly.

3 Analysis

3.1 Data Exploration

The M4 dataset consists of 100,000 univariate time series from six domains, divided in six different frequencies. They were chosen from a database of more than 900,000

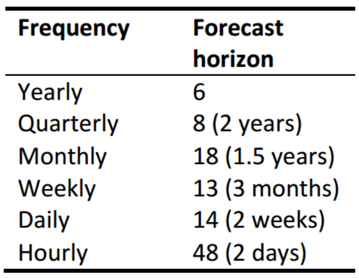
M4 Data - Overview

Frequency	Micro	Industry	Macro	Finance	Demographic	Other	Total
Yearly	6,538	3,716	3,903	6,519	1,088	1,236	23,000
Quarterly	6,020	4,637	5,315	5,305	1,858	865	24,000
Monthly	10,975	10,017	10,016	10,987	5,728	277	48,000
Weekly	112	6	41	164	24	12	359
Daily	1,476	422	127	1,559	10	633	4227
Hourly						414	414
Total	25,121	18,798	19,402	24,534	8,708	3,437	100,000

Table 1: M4 data by domains and frequencies.

time series. Each series consists of a date and a respective observation for that date, where the frequency determines the time between two observations. In this dataset, all frequencies are equidistant which means that the time between two observations is constant. Common characteristics of time series include seasonal patterns, cyclic behavior, as well as trends that the model has to capture. Given the behavior of the series, the method has to extrapolate the series into the future. The number of future observations (forecast horizon) that had to be forecasted was given: A key challenge for the prediction method is the high diversity of the data. Some data

M4 Data - Forecast Horizons



Frequency	Forecast horizon
Yearly	6
Quarterly	8 (2 years)
Monthly	18 (1.5 years)
Weekly	13 (3 months)
Daily	14 (2 weeks)
Hourly	48 (2 days)

Table 2: Forecast horizons by frequency of the data.

exhibit cyclical behavior, whereas other series exhibit up-/ downward trends, or varying seasonal patterns. Capturing the behavior accurately for all series is the goal of time series forecasting.

3.2 Exploratory Visualization

The dataset is highly diverse with respect to domain and frequency. For each series there exists a training set as well as extended test set. The test set consists of the train set and the additional, previously unknown observations to be forecasted. A randomly chosen example series from the hourly M4 dataset is depicted below:

M4 Hourly - Series #338

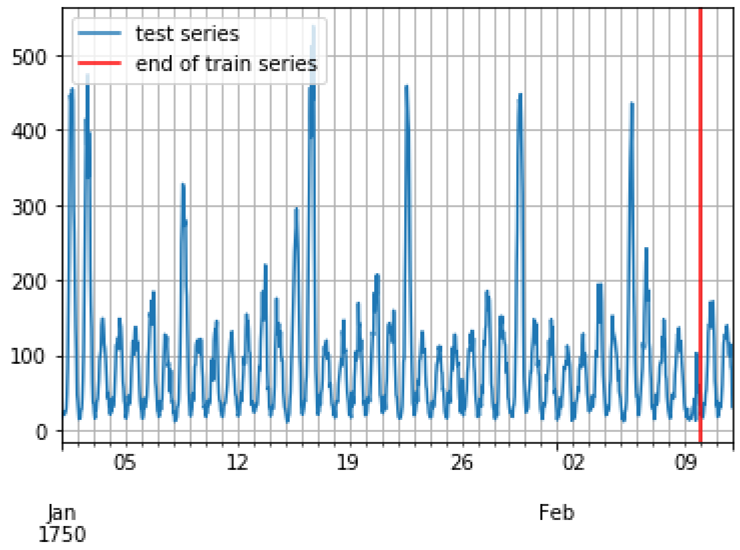


Figure 1: Example series over time. The vertical red line indicates where the training data ends and the test data begins.

3.3 Algorithms and Techniques

The first of the three algorithms is the **Deep Autoregressive Recurrent Neural Network (DeepAR)** method, introduced by Salinas et al. (2017). In the AWS Sageaker for ML in the cloud, it is the sole built-in time series prediction algorithm. DeepAR uses autoregressive features that are used as the inputs to a recurrent neural network. The RNN uses LSTM cells as default but can also be changed to GRU cells. Previous observations are utilized as inputs which makes this algorithm autoregressive. Autoregressive features are included based on the frequency of the data.

Rangapuram et al. (2018) introduced **Deep State Space Models (DeepState)**. DeepState combines state space models (SSM) with a recurrent neural network architecture that calculates the optimal parameters for the SSM. The SSM is applied locally to the individual time series. The parameters are determined by a RNN which is trained jointly on all available series. Hence, this algorithm combines a local with a global approach.

Furthermore, I included the **Deep Factor - RNN** by Wang et al. (2019). This hybrid algorithm uses any local method specific to each time series in the dataset as well as a global RNN trained on the entire data. For now, the only local method that is available is another RNN. Hence, the model combines a local RNN with a global RNN (**DF-RNN** in the following). Main differences between the algorithms:

- **DeepAR** - Pure RNN that uses autoregressive features as inputs.
- **DeepState** - Uses global RNN to determine the best parameters of the local SSM.
- **DF-RNN** - Hybrid of a local method (here: RNN) applied to individual time series that is combined with a global RNN that is trained jointly on all time series.

The training process for all three algorithms uses a fixed window approach. During training, batches of time series windows in the training data are fed to the algorithm. Each time series window is a sliced version of the entire series starting at different points of the series and has length of the training range plus prediction range. Here, the prediction range corresponds to the forecasting horizon of our model. Stochastic Gradient Descent (SGD) is used and during the training process the epoch with the smallest loss is chosen for the final model.

3.4 Benchmarks

In addition to providing a rich dataset across frequencies and domains, a key contribution of the M4 competition is the provision of a rich set of benchmark methods. Not only the metrics of the competition winners but also results for commonly used methods, in particular ARIMA and ES, are provided in Hyndman et al. (2019). In the discussion of the results I will focus on the relative performance of the RNN-based methods with respect to:

1. Seasonally adjusted naive predictions (Naive2)
2. Traditional methods (ARIMA and ES) and combination benchmark (Comb)
3. Overall winning method of Smyl (2019), runner-up by Montero-Manso et al. (2019) as well as the winning method of the frequency under consideration.

4 Methodology

4.1 Data Preprocesing

The data has to be prepared to be a valid input to the algorithms. Inputs are required to be in a **JSON Lines**-like format which in python is essentially a dictionary of dictionaries (nested dictionary). Inputs must contain at least the following keys with their respective content:

- `‘start‘` - Timestamp with the format YYYY-MM-DD HH:MM:SS and the respective frequency of the data
- `‘target‘` - An array of floats or integers that represents the target series of interest.

An example input file takes the form

```
# get first entry in dataset.train
entry = list(dataset.train)[0]

# first entry
entry
>>> {'start': Timestamp('1750-01-01 00:00:00', freq='H'),
  'target': array([605., 586., 586., ..., 521.], dtype=float32)
```

Additional keys are optional but often helpful in improving forecast accuracy. Here, the M4 competition does not provide additional covariates to the data.

4.2 Implementation

To obtain the results of the paper I used the **gluonts** library that is built on the deep learning library **MXNet**. For all three algorithms I proceeded in a similar way.

First, I implemented a function that takes in the following arguments:

- data - name of the built-in data in gluonts that was used
- seed - a seed that fixes the random numbers. Using the same seed results can be reproduced.
- epochs - Number of epochs a model is trained
- batches - Number of batches per epoch that are used. Each epoch the algorithm is fed using this many training examples.

```
def deepar(data="m4_weekly", seed=42, epochs=100, batches=50):

    dataset = get_dataset(data, regenerate=False)
    mx.random.seed(seed)
    np.random.seed(seed)
    ...
```

Applying this function on a dataset yields results aforementioned accuracy measures:

```
res = deepar(data="m4_weekly", seed=42, epochs=1, batches=50)
pprint(res)

>>> {'MASE': 5.54378962,
 'MSIS': 54.72282354,
 'epochs': 1,
 'sMAPE': 0.12889508,
 'seed': 42,
 'wQuantileLoss[0.5]': 0.09491686,
 'wQuantileLoss[0.9]': 0.07740071}
```

A key finding was the instability of the algorithms. Applying the same algrithm with the same hyperparameters (i.e. epochs and batches) on one identical dataset leads to completely different results due to variability in the random numbers that are used. Hence, I had to implement a way to make the results reproducible by including seeds.

Also, in order to cancel out some of the variation in the results between iterations, I needed to repeat the experiments several times. For this, I wrote a loop that applies the respective function over a range of seeds and repeated the experiments holding the epochs and number of batches constant. The results show the median of three trials.

```
results = []

data="m4_monthly"
epochs=100
batches=50

if __name__ == "__main__":
    for i in range(42, 45):
        print("Seed:", i)
        res = deepar(data, seed=i, epochs, batches)
        pprint(res)
        results.append(res)
```

This experimental arrangement ensures that the results are reproducible and meaningful by using seeds and several trials.

4.3 Refinement

A problem for an efficient implentation of the algorithms is the diverse nature of the data and the many different options of hyperparameters. The number of series within the four datasets range from 359 (M4 weekly) to 48,000 series (M4 monthly). Moreover, the data stems from different domains. There is seemingly no relation between the data. Aforementioned cross-learning by exploiting common patterns and dependencies seems difficult in this setting. Also, these algorithms allow many arguments to be determined such as prediction length of the forecasting horizon, context length to consider, number of RNN layers, number of RNN cells for each layer, or dropout rate just to mention a few of them. Besides prediction length which is fixed due to the restrictions of the competiton, there are many flexible parts for each forecasting problem to be considered.

A common solution to this problem is to employ hyperparameter optimization (HPO) techniques, such as grid search or random search. These HPO techniques bascially run different variations of the same algorithm to determine the best hyperparameters. This is not a viable option here. Some the models run for many hours per trial and running different variations multiplies computational costs. Due to limited resources I decided to start with the default values which should already deliver reasonable results and only adjusting the number of epochs, number of batches per epoch, and the seed of the random numbers.

5 Results

This section describes the results per frequency, starting from the lowest (hourly) to the highest one (monthly). For every combination of algorithm, data, epochs, and batches, only the median result of three trials is included in the table. Trials start with seed 42 because it is "the ultimate answer to life, the universe, and everything..." (Adams, 2005). Note that DeepState models are computationally more expensive. Hence, I limited experimentation with DeepState to an acceptable minimum as some models took more than 13 hours of CPU time per trial. Because this algorithm is more stable than the others it shouldn't alter the conclusions on its accuracy.

5.1 Model Evaluation and Validation

M4 Hourly

The hourly subset of the M4 data contains 414 time series. Given forecast horizon (i.e. the number of future observations) was 48, which corresponds to two days.

Table 5.1 illustrates the performance of aforementioned algorithms, respective benchmarks, and winning methods applied on the hourly data of the M4 data.

It seems that DF-RNN is not able to capture hourly data at all. Comparing either its MASE or sMAPE to the seasonally adjusted naive forecast (Naive2) shows how bad it works. Note that the performance is even worse than a naive forecast that simply takes the previous value as forecast (value not shown in the table).

With respect to Naive2 and the combination benchmark both models of DeepAR and DeepState show a higher accuracy. Moreover, they are also able to perform more accurate than ETS. However, the traditional ARIMA method that is based on modeling the autoregressive behavior of time series, outperforms both models with ease.

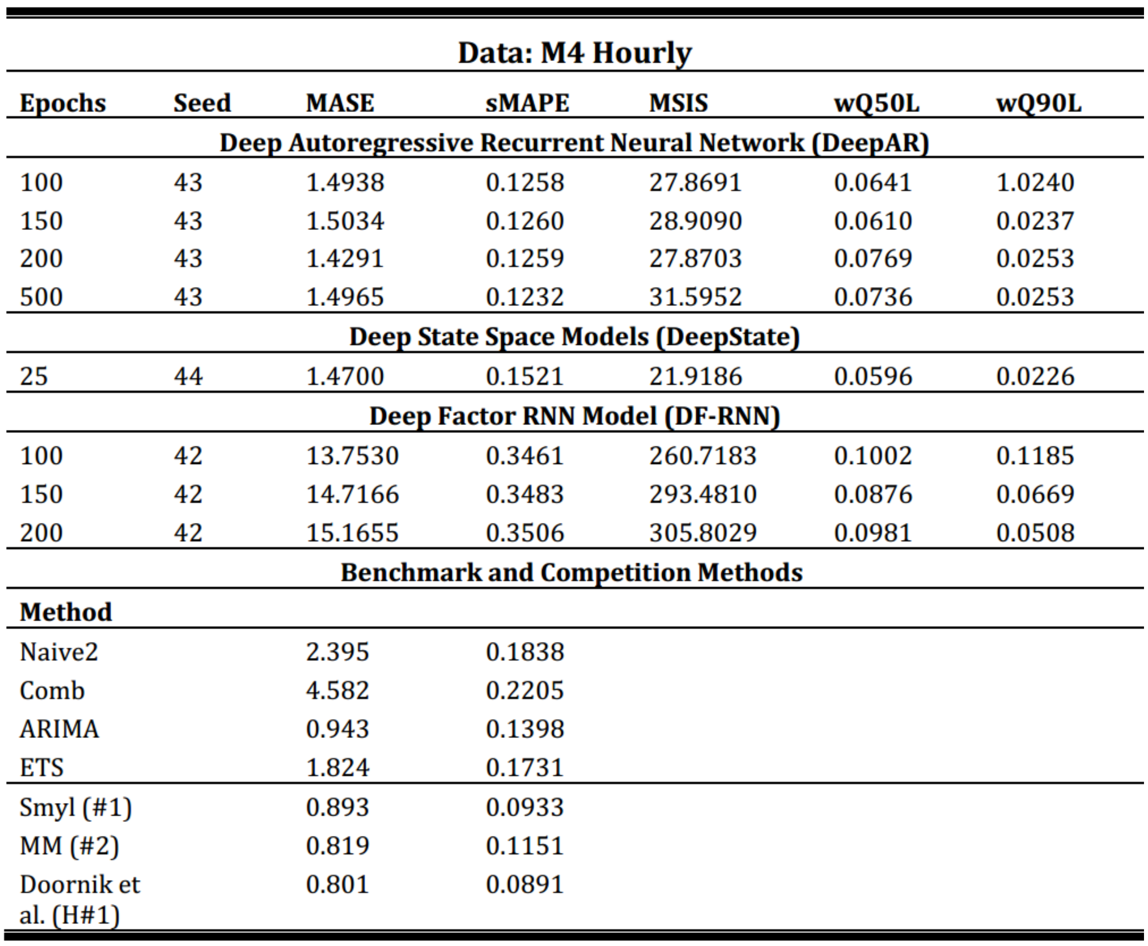
Additionally, neither of the considered three algorithms comes close to the overall winning method by Smyl which is also based on recurrent neural networks. In addition, Montero-Manso (2019) and the best hourly method of Doornik et al. capture hourly data more precisely than the winning method.

DeepAR and DeepState perform similarly in this setting. Their performance however is definitely not impressive. Note that this is a rather small dataset and model converge rather quick during training.

M4 Daily

Next, I will present the results for the M4 daily dataset which consists of 4227 series. The task is to predict the next 14 observations, i.e. 2 weeks.

M4 Hourly - Results



Data: M4 Hourly						
Epochs	**Seed**	**MASE**	**sMAPE**	**MSIS**	**wQ50L**	**wQ90L**
Deep Autoregressive Recurrent Neural Network (DeepAR)						
100	43	1.4938	0.1258	27.8691	0.0641	1.0240
150	43	1.5034	0.1260	28.9090	0.0610	0.0237
200	43	1.4291	0.1259	27.8703	0.0769	0.0253
500	43	1.4965	0.1232	31.5952	0.0736	0.0253
Deep State Space Models (DeepState)						
25	44	1.4700	0.1521	21.9186	0.0596	0.0226
Deep Factor RNN Model (DF-RNN)						
100	42	13.7530	0.3461	260.7183	0.1002	0.1185
150	42	14.7166	0.3483	293.4810	0.0876	0.0669
200	42	15.1655	0.3506	305.8029	0.0981	0.0508
Benchmark and Competition Methods						
Method						
Naive2		2.395	0.1838			
Comb		4.582	0.2205			
ARIMA		0.943	0.1398			
ETS		1.824	0.1731			
Smyl (#1)		0.893	0.0933			
MM (#2)		0.819	0.1151			
Doornik et al. (H#1)		0.801	0.0891			

Table 3: Results for hourly M4 data.

M4 Weekly

Among the M4 subsets by frequency, the M4 weekly dataset has the fewest with 359 individual series. Forecast range is 13 which is roughly equivalent to three months or one quarter.

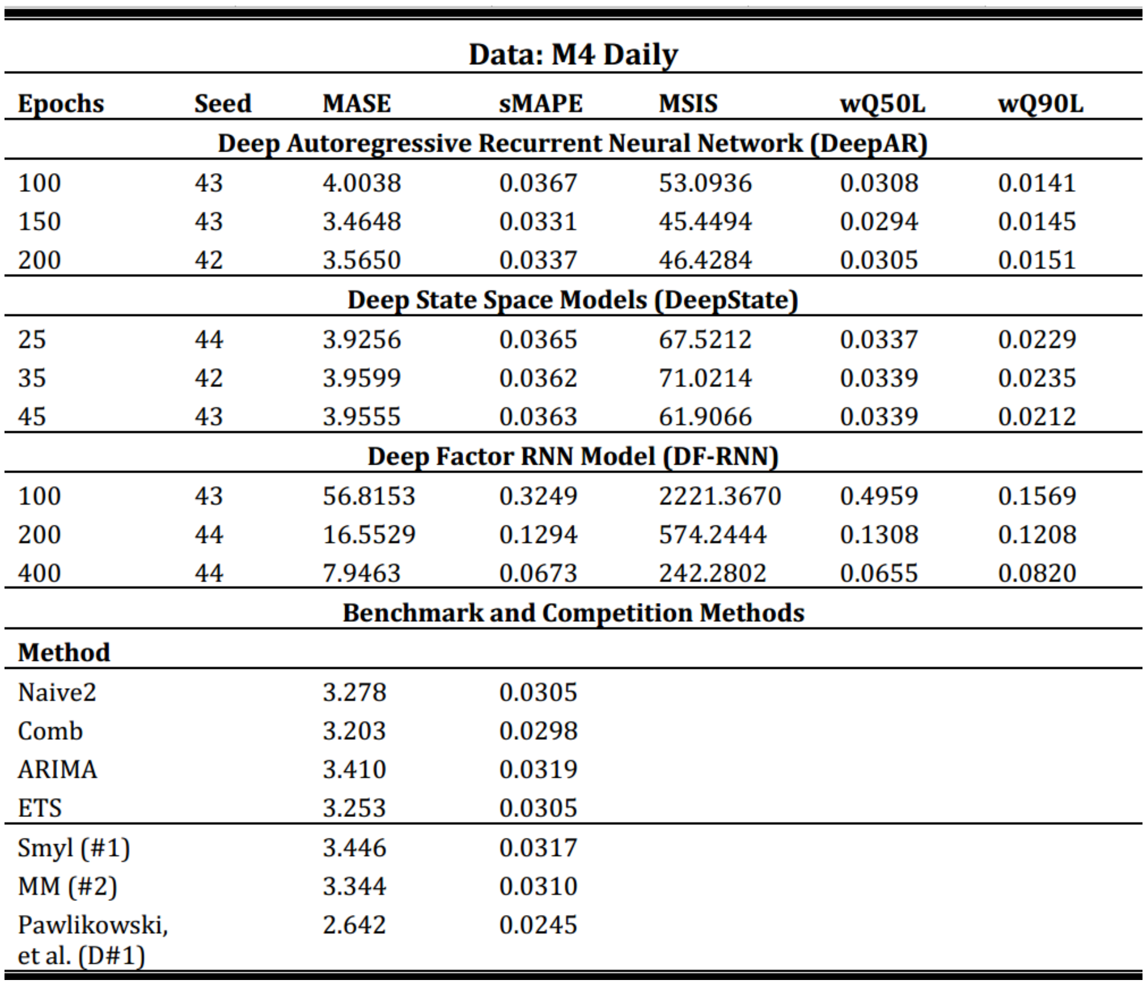
M4 Monthly

The monthly data of the M4 competition contains 48,000 time series and is the largest among all frequencies.

5.2 Justification

The main question of this project is to whether RNN-based algorithms may be a valuable tool in time series forecasting.

M4 Daily - Results



Data: M4 Daily						
Epochs	**Seed**	**MASE**	**sMAPE**	**MSIS**	**wQ50L**	**wQ90L**
Deep Autoregressive Recurrent Neural Network (DeepAR)						
100	43	4.0038	0.0367	53.0936	0.0308	0.0141
150	43	3.4648	0.0331	45.4494	0.0294	0.0145
200	42	3.5650	0.0337	46.4284	0.0305	0.0151
Deep State Space Models (DeepState)						
25	44	3.9256	0.0365	67.5212	0.0337	0.0229
35	42	3.9599	0.0362	71.0214	0.0339	0.0235
45	43	3.9555	0.0363	61.9066	0.0339	0.0212
Deep Factor RNN Model (DF-RNN)						
100	43	56.8153	0.3249	2221.3670	0.4959	0.1569
200	44	16.5529	0.1294	574.2444	0.1308	0.1208
400	44	7.9463	0.0673	242.2802	0.0655	0.0820
Benchmark and Competition Methods						
Method						
Naive2		3.278	0.0305			
Comb		3.203	0.0298			
ARIMA		3.410	0.0319			
ETS		3.253	0.0305			
Smyl (#1)		3.446	0.0317			
MM (#2)		3.344	0.0310			
Pawlikowski, et al. (D#1)		2.642	0.0245			

Table 4: Results for M4 Daily data.

6 References

[1] Adams, Douglas. "The Hitchhiker's Guide to the Galaxy". Del Rey Books. 2005

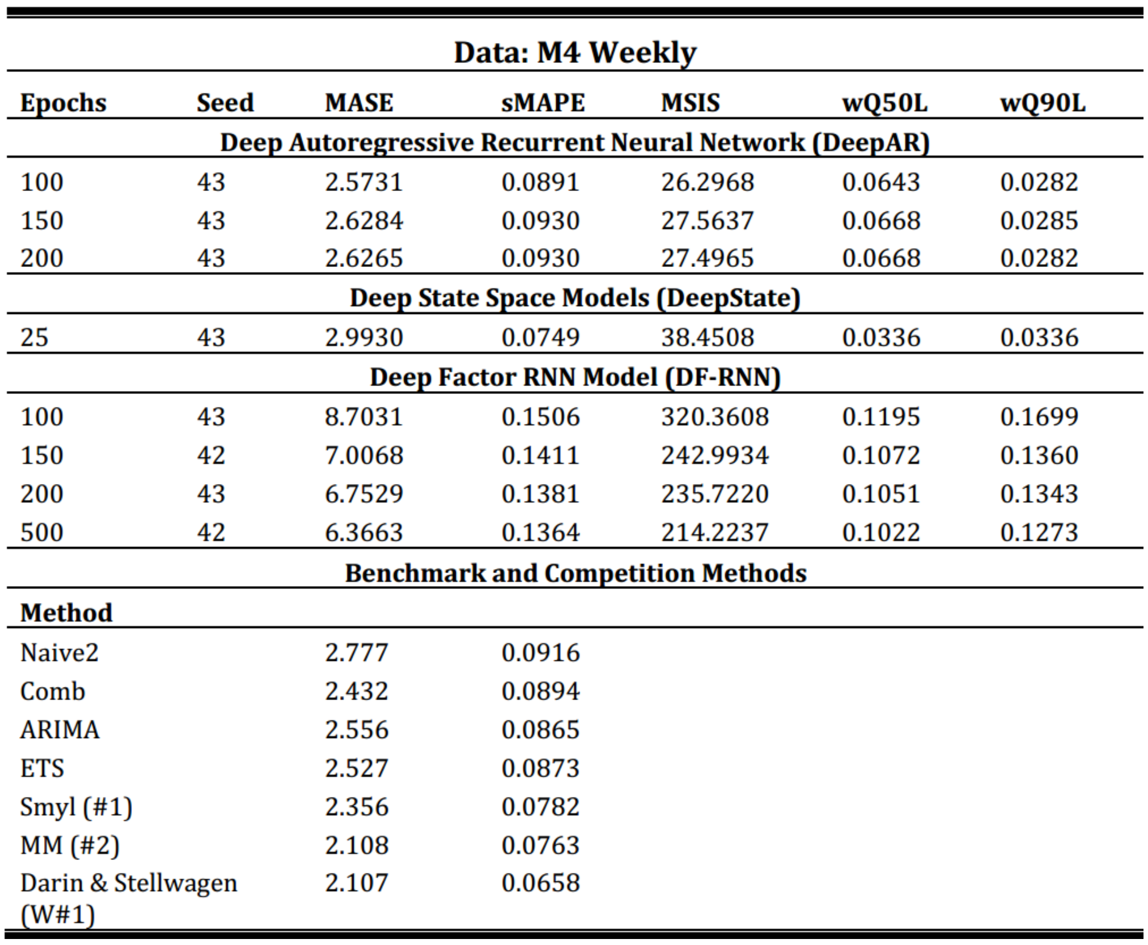
[2] Hyndman, Rob J. "A brief history of forecasting competitions." *International Journal of Forecasting (2019).*

[3] Hyndman, Rob J., and Anne B. Koehler. "Another look at measures of forecast accuracy." *International journal of forecasting 22.4 (2006): 679-688.*

[4] Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "Statistical and Machine Learning forecasting methods: Concerns and ways forward." *PloS one 13.3 (2018): e0194889.*

[5] Makridakis, Spyros, Evangelos Spiliotis, and Vassilios Assimakopoulos. "The

M4 Weekly - Results



Data: M4 Weekly						
Epochs	**Seed**	**MASE**	**sMAPE**	**MSIS**	**wQ50L**	**wQ90L**
Deep Autoregressive Recurrent Neural Network (DeepAR)						
100	43	2.5731	0.0891	26.2968	0.0643	0.0282
150	43	2.6284	0.0930	27.5637	0.0668	0.0285
200	43	2.6265	0.0930	27.4965	0.0668	0.0282
Deep State Space Models (DeepState)						
25	43	2.9930	0.0749	38.4508	0.0336	0.0336
Deep Factor RNN Model (DF-RNN)						
100	43	8.7031	0.1506	320.3608	0.1195	0.1699
150	42	7.0068	0.1411	242.9934	0.1072	0.1360
200	43	6.7529	0.1381	235.7220	0.1051	0.1343
500	42	6.3663	0.1364	214.2237	0.1022	0.1273
Benchmark and Competition Methods						
Method						
Naive2		2.777	0.0916			
Comb		2.432	0.0894			
ARIMA		2.556	0.0865			
ETS		2.527	0.0873			
Smyl (#1)		2.356	0.0782			
MM (#2)		2.108	0.0763			
Darin & Stellwagen (W#1)		2.107	0.0658			

Table 5: Results for weekly M4 data.

M4 competition: 100,000 time series and 61 forecasting methods." *International Journal of Forecasting (2019).*

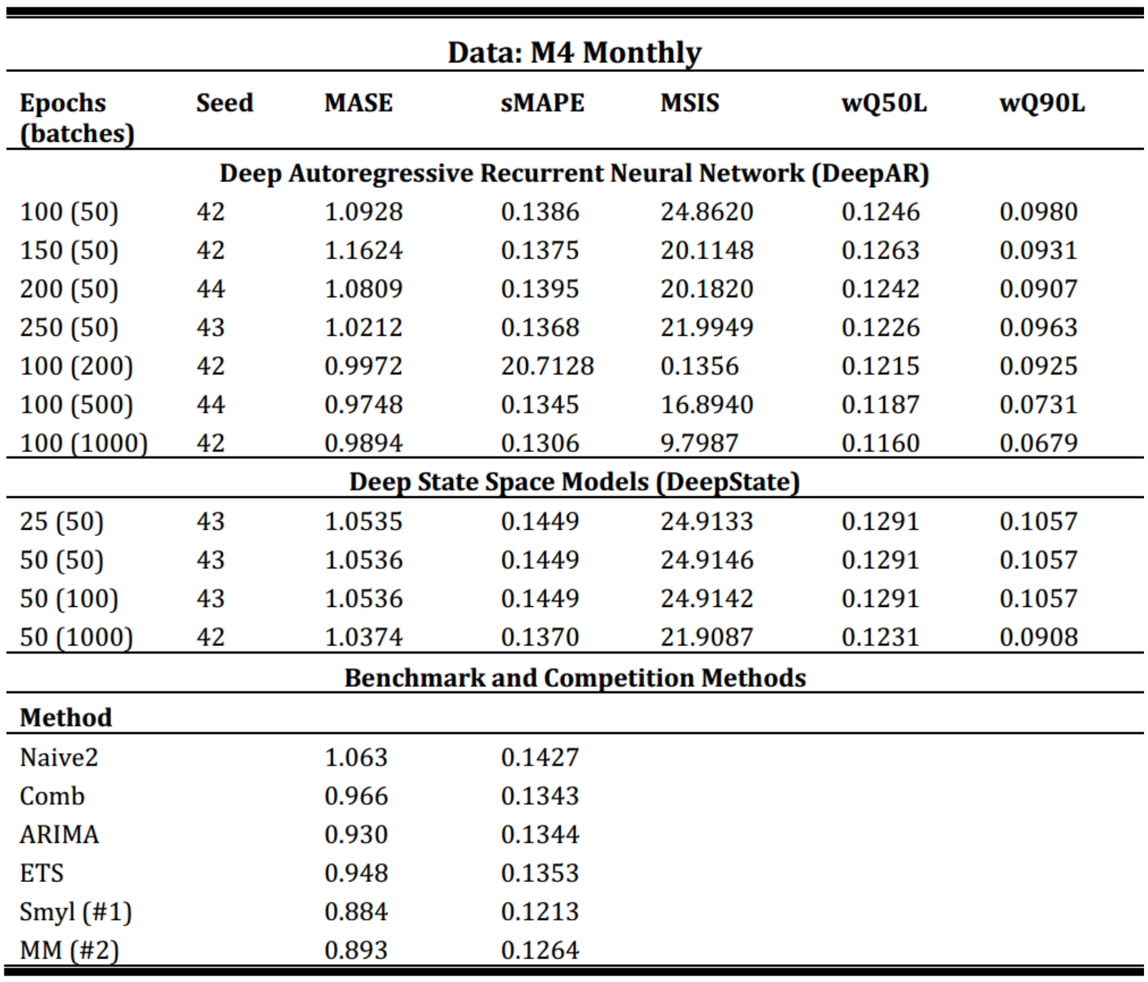
[6] Montero-Manso, Pablo, et al. "FFORMA: Feature-based forecast model averaging." *International Journal of Forecasting (2019).*

[7] Rangapuram, Syama Sundar, et al. "Deep state space models for time series forecasting." *Advances in Neural Information Processing Systems.* 2018.

[8] Salinas, David, Valentin Flunkert, and Jan Gasthaus. "DeepAR: Probabilistic forecasting with autoregressive recurrent networks." *arXiv preprint arXiv:1704.04110 (2017).*

[9] Smyl, Slawek. "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting." *International Journal of Forecasting (2019).*

M4 Monthly - Results



Data: M4 Monthly						
Epochs (batches)	**Seed**	**MASE**	**sMAPE**	**MSIS**	**wQ50L**	**wQ90L**
Deep Autoregressive Recurrent Neural Network (DeepAR)						
100 (50)	42	1.0928	0.1386	24.8620	0.1246	0.0980
150 (50)	42	1.1624	0.1375	20.1148	0.1263	0.0931
200 (50)	44	1.0809	0.1395	20.1820	0.1242	0.0907
250 (50)	43	1.0212	0.1368	21.9949	0.1226	0.0963
100 (200)	42	0.9972	20.7128	0.1356	0.1215	0.0925
100 (500)	44	0.9748	0.1345	16.8940	0.1187	0.0731
100 (1000)	42	0.9894	0.1306	9.7987	0.1160	0.0679
Deep State Space Models (DeepState)						
25 (50)	43	1.0535	0.1449	24.9133	0.1291	0.1057
50 (50)	43	1.0536	0.1449	24.9146	0.1291	0.1057
50 (100)	43	1.0536	0.1449	24.9142	0.1291	0.1057
50 (1000)	42	1.0374	0.1370	21.9087	0.1231	0.0908
Benchmark and Competition Methods						
Method						
Naive2		1.063	0.1427			
Comb		0.966	0.1343			
ARIMA		0.930	0.1344			
ETS		0.948	0.1353			
Smyl (#1)		0.884	0.1213			
MM (#2)		0.893	0.1264			

Table 6: Results for monthly M4 data.

[10] Spiliotis, Evangelos, et al. "Are forecasting competitions data representative of the reality?." *International Journal of Forecasting (2019).*

[11] Wang, Yuyang, et al. "Deep Factors for Forecasting." *arXiv preprint arXiv:1905.12417 (2019).*