M4 Competition Implementations*

*Note: All authors contributed equally to this project

Eustathios Kotsis
StudentID: 7115152200008
efstathkot@di.uoa.com

Michael Darmanis
StudentID: 7115152200004
mdarm@di.uoa.com

Nektarios Christou
StudentID: 7115152200002
nekchristou@di.uoa.com

Vasilios Venieris
StudentID: 7115152200017
vvenieris@di.uoa.com

***Abstract*—This project dives into some of the methodologies (and benchmarks) presented in the M4 Forecasting Competition. By implementing and juxtaposing these models, the aim was to highlight their respective merits and practical implications. Experimental results, though somewhat lacking, agreed with the competition's findings [2]; that is, pure machine learning methods, despite their potential, did not outperform simpler statistical methods or combinations thereof. Hybrid methods, like ES-RNN, which combine the strengths of both worlds, showed the most promise.**

***Index Terms*—forecasting, es-rnn, cnn, mlp, arima, naïve-techniques**

I. Introduction

The domain of time series forecasting, pivotal across various sectors, continually witnesses the emergence of refined methodologies. The M4 Forecasting Competition [1], a benchmark in this sphere, has introduced a myriad of approaches, each addressing the challenges of time series data in its unique manner.

This effort attempts to detail the intricacies of forecasting models such as Convolutional Neural Networks (CNN), Multi-Layer Perceptrons (MLP), Exponential Smoothing with Recurrent Neural Networks (es-rnn), and traditional statistical methods, including ARIMA and Naïve techniques.

The subsequent sections will summarise each implementation, showcasing the methodology, operational mechanics, and comparative performance. Possible reasons for why some models performed better than others will be discussed.

II. Forecasting Implementations

The M4 competition [1] utilised the average of two of the most widely recognized accuracy measures, which are jointly referred to as the overall weighted average (OWA). This approach enhanced the evaluations' objectivity. The chosen measures for were the symmetric mean absolute percentage error (sMAPE) and the mean absolute scaled error (MASE). The methods' performances in terms of the accuracy of PFs can be seen in Table I

TABLE I
Overall M4-Performances of Implementations [1]

Method	OWA	sMAPE	MASE
ES-RNN	0.899	12.555	0.898
CNN	0.920	13.434	0.920
MLP Benchmark	0.920	13.434	0.920
Naive 1	1.000	14.073	1.000
Naive 2	0.920	13.434	0.920

It is important to note that our implementations were not direct replications of the original methodologies and were not subjected to testing across all available datasets. As such, there were variations in performance outcomes when compared to the original methods.

A. Convolutional Neural Networks

The primary objective here was to train and predict using the original Btrotta models [7] for forecasting. The original models were designed to capture trends and seasonalities of time series for different frequencies without manually extracting characteristics. The convolutional kernels are believed to capture seasonality patterns, with 3 filters used for each convolutional layer. Different models were constructed for each frequency, horizon time step, and training length.

1) Preprocessing and Experiments: The original implementation suggests that recent time-series-observations are crucial. Training lengths were, therefore, selected based on the end of the training data and were multiples of all the frequencies that the series might exhibit. For series with insufficient data, synthetic data was generated to maintain periodic patterns. The standard approach of subtracting the mean and dividing by the standard deviation was employed for standardisation. For series with not enough samples, the series were duplicated and shifted backward one step, creating a series of the same length. Indicative results were those of quarterly and yearly predictions which yielded *MASE: 1.23, 2.81* and *sMAPE: 9.04, 10.62* respectively.

In most cases, the newly designed models performed similarly or better than the original models. This suggests that the naive forecast might be modeling some of the series' underlying patterns. For the yearly frequencies, the new models, which applied pooling and convolutional filters, performed

Identify applicable funding agency here. If none, delete this.

similarly or better, indicating potential patterns in the yearly series. The monthly frequency benefited from the inclusion of naive forecasts in the results. For the daily frequency, a training length of 364 days was added, which seemed to enhance performance, suggesting that a longer history might help identify potential periodic events.

Multi-step convolutional models were also trained on varying lengths but did not yield optimal results.

B. Multi-Layer Perceptrons

The model here, based on the MLP Benchmark [8], consisted of an input layer, a hidden layer, and an output layer. Early stopping was employed during the training phase to avert overfitting. The model's efficacy was gauged using performance metrics like Mean Squared Error (MSE) and Mean Absolute Error (MAE).

1) Preprocessing and Experiments: Before feeding the data into the MLP model, several preprocessing steps were undertaken. The time series data was initially loaded and processed to handle any missing values. Trends and seasonality, especially for non-yearly frequencies, were removed. Essential time-centric details, such as the days of the week for hourly datasets, were integrated using one-hot encoding techniques. Furthermore, the dataset underwent scaling to achieve normalisation, and subsequently, input sequences paired with their corresponding target values were generated.

While the training loss and MAE showed promising results in the initial epoch, they began to decline in the following epochs, hinting at potential overfitting. This observation was further verified by the validation metrics, which also exhibited a deteriorating trend as training advanced. The test evaluation outcomes mirrored the validation set's trend, suggesting that the model might not have generalised effectively. Varying model architectures were experimented with while and hyperparameters were tuned. Implementation of ”time embeddings” or ”time encoding,” was also tried. This last technique encoded time-related information into a continuous vector, capturing cyclic patterns like daily, weekly, or yearly variations, in the hope of boosting forecasting accuracy.

Performing all the above on the hoursly dataset, yielded an *OWA* score of 0.70.

C. Exponential Smoothing with Recurrent Neural Networks

The algorithm[1] can be divided into two distinct layers: a preprocessing layer that uses exponential smoothing and a LSTM layer that updates the per-series parameters of the Holts-Winter model.

In the pre-processing layer, Holts-Winter exponential smoothing [5] with multiplicative seasonality and trend is applied via

$$l_t = \alpha(\frac{y_t}{s_{t-m}}) + (1-\alpha)l_{t-1}b_{t-1} \quad (1)$$

$$b_t = \beta(\frac{l_t}{l_{t-1}}) + (1-\beta)b_{t-1} \quad (2)$$

$$s_t = \gamma\frac{y_t}{l_{t-1}b_{t-1}} + (1-\gamma)s_{t-m} \quad (3)$$

$$\hat{y}_{t+h} = l_t b_t^h s_{t-m+h_m^+} \quad (4)$$

where l is a state variable for level, b is a state variable for trend and s is a multiplicative seasonality coefficient. α, β and γ are smoothing coefficients between zero and one. An h step forecast is given by Equation 4.

Since the model no longer considers local linear trend, Equation 2 is replaced by an RNN from the model as follows:

$$\hat{y}_{t+1...t+h} = RNN(X_t) * l_t * s_{t+1...t+h} \quad (5)$$

$$x_i = \frac{y_i}{l_t s_i} \quad (6)$$

X_t is a vector of normalized, de-seasonalised features of which a scalar component x_t is calculated via Equation 6.

The performed implementation was not the same as that of Slawek's [3]. Particularly, it did not handle many time series at once, the L1-metric was used as loss function, and a GRU layer was used instead of the LSTM ones.

1) Preprocessing and Experiments: During the data preprocessing, trailing NaN-values were removed while the remaining missing values in the series were interpolated using a linear method to ensure continuity. Additionally random shifts in the input data were performed to generate more diverse training samples.

Experiments were performed only on the hourly dataset yielding results subpar of the original implementation's (*OWA: 1.67, MASE: 1.20, sMAPE: 15.67.*This is to be expected. Slawek's [3] architecture was much more complex, trained in the entirety of the dataset, at once, and had additional components (LSTM and dialated RNN layers) that captured time series patterns better. Slawek also used ensembling techniques to achieve his competition-winning results.

D. Naïve 1, 2 & S and ARIMA

Four models, in their varying degrees of simplicity and complexity, served as a benchmark against which other models in the project were compared. The Naïve models, by virtue of their design, operate on the principle that recent historical data is the most indicative of future outcomes [6]. This approach, while straightforward, can often capture patterns and trends in time series data without the need for extensive modeling or computations.

1) Preprocessing and Experiments: The three Naïve models were grounded in a fundamental assumption: the most recent data points held the highest predictive power [6]. Despite their simplicity, they surprisingly outperformed the more sophisticated ARIMA(p,d,q) for several data series. One plausible explanation for this could be the introduction of persistent upward or downward trends when consecutive

[1] the hybrid ES-RNN algorithm is discussed in detail in a blog post by Slawek himself [4]

differences were taken into account. Such trends might not always align with the actual data, which could end or reverse its direction, leading to increased errors. Additionally, any missing data within the series was interpolated to ensure continuity and accuracy.

While Naive S and 2 retained the core principle of relying on recent values, they introduced slight complexities. Naive S, rather than just replicating the last known data point, replicated a set period 'S' of them. This modification led to a notable improvement in performance, with *sMAPE* reductions ranging between 10-37% compared to Naive 1. On the other hand, Naive 2 operated similarly to Naive 1 but necessitated some pre-processing steps. However, its performance enhancements over Naive 1 were modest. Intriguingly, for approximately 30% of the yearly data series, Naive 2's *sMAPE* lagged behind that of Naive 1.

III. Findings and Further Discussion

Both CNN and MLP are categorised under machine learning methods. As per the M4-competition results [2], the six pure ML methods submitted in the M4 generally underperformed, with none surpassing the accuracy of Comb and only one outperforming Naïve2. This can be attributed to the nature of machine learning methods, particularly deep learning ones like CNNs, which require a substantial amount of data for training. The time series data in forecasting competitions might not always be sufficient for such models, and there is always a risk of these models overfitting if they are not adequately regularised.

Conversely, the ES-RNN hybrid approach, which emerged as the competition's winner, stood out and caught everyone by surprise. This approach, which seamlessly integrated both statistical and ML features, not only produced the most accurate forecasts but also the most precise PIs, outperforming the combination benchmark by nearly 10%.

This hybrid method effectively integrated the advantages of both statistical methods and machine learning. This approach assimilated information from individual series as well as the entire dataset, capitalising on data in a hierarchical manner, according to Slawek [4]. Given the sequential design of RNNs, they were inherently suitable for time series forecasting.

In conclusion, while numerous advanced methods struggled to top the benchmarks, only one pure ML method managed to be more accurate than Naïve2. This observation aligns with expectations. Naive methods, renowned for their straightforwardness, frequently set the standard in forecasting competitions. Indeed, it happens not infrequently data is full of noise, and intricate models may not consistently hold an edge over these statistical benchmarks.

Acknowledgments

This report was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. A template that can be used to format documents with this look and feel has been released under the IEEE License, and can be found online at https://www.overleaf.com/latex/templates/ieee-conference-template/grfzhhncsfqn.

References

[1] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: 100,000 time series and 61 forecasting methods," Int. J. Forecasting, vol. 36, no. 1, pp. 54–74, 2020.

[2] S. Makridakis, E. Spiliotis, and V. Assimakopoulos, "The M4 Competition: Results, findings, conclusion and way forward," Int. J. Forecasting, vol. 34, no. 4, pp. 802–808, 2018.

[3] S. Smyl, "A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting," Int. J. Forecasting, vol. 36, no. 1, pp. 75–85, 2020.

[4] [Online] Available: https://www.uber.com/en-GR/blog/m4-forecasting-competition/. "M4 Forecasting Competition: Introducing a New Hybrid ES-RNN Model."

[5] P. R. Winters, "Forecasting sales by exponentially weighted moving averages," Management science, vol. 6, no. 3, pp. 324–342, 1960.

[6] R. J. Hyndman and G. Athanasopoulos, Forecasting: principles and practice. OTexts, 2018.

[7] [Online] Available: https://github.com/Mcompetitions/M4-methods/tree/master/211-btrotta. "Convolutional Neural Networks for Forecasting."

[8] [Online] Available: https://github.com/Mcompetitions/M4-methods/blob/master/ML_benchmarks.py. "MLP Benchmark."