Global Models, Zero-shot Forecasting and Fully Connected Neural Networks

Abstract

Recent research has shown that global time series models are reasonable both from a practical perspective and a theoretical perspective. In this paper, we consider one of the simplest neural network models possible, the multilayer perceptron (MLP), and show that a carefully trained and selected MLP performs on par with state-of-the-art models while requiring far fewer parameters. In addition, we show that such a model also performs favourably in the zero-shot forecasting scenario, where the time series in the test set is completely unseen when training the model. Our results indicate that representation learning may be a fruitful endeavour in the forecasting setting.
Keywords: time series, global models, representation learning, forecasting, neural networks

1. Introduction

Forecasting remains to be one of the most important challenges for statistical models and machine learning models. Knowing future outcomes of a process may improve safety, efficiency and profitability for systems, companies or societies associated with the process. Finding robust and efficient ways to forecast time series and understanding their pros and cons is consequently associated with significant value for both businesses and policymakers.

In this article, we explore the properties of global time series models created using fully connected neural networks. We show that despite the conceptual simplicity of fully connected neural networks, these models are competitive with more complex models, such as the state-of-the-art N-BEATS model (Oreshkin, Carpov, Chapados & Bengio, 2020b). Furthermore, we investigate their performance in the "zero-shot" forecasting setting, namely a situation where a pre-trained model is used to forecast unseen time series. Automatic forecasting or AutoML for time series is an area that is getting more attention these days, and our experiments on zero-shot learning provide hope in this area. In many modeling setups, ensembles are used to produce better forecasts. In this article, we explore the effects of these ensembles and show that the best member of a simple fully connected model performs on par with a much more complex ensemble while having far fewer

parameters than their ensemble counterpart.

In our experiments, we use the well-known M4 dataset (Makridakis, Spiliotis & Assimakopoulos, 2020) for standard forecasting experiments and the FRED dataset (Federal Reserve Bank of St. Louis, 2021) for zero-shot experiments.

2. Related Work

In the forecasting literature, we distinguish between what is called local and global models, as described in Montero-Manso & Hyndman (2021). The methodology described in this article belongs to the group of global forecasting models. In Section 4, we also distinguish between global and *almost* global models. In Montero-Manso & Hyndman (2021) they provide a theoretical framework for analyzing global and local models, and show in which sense they are comparable. They also show experimentally that choosing the right number of lags in an auto-regressive global model is important to remove aliasing effects, and create several strong baselines for variants of global forecasting models for the M4 dataset. While their best performing variants include some extra covariates (such as frequency information) to produce the best models, our experiments indicate that these features are not needed to achieve comparable performance. N-BEATS (Oreshkin et al., 2020b) is a global forecasting model that makes adaptations to the MLP in order to facilitate time series in a more natural way, and also some optional modifications that makes the model more interpretable and explainable. N-BEATS is to the best of our knowledge the model that performs best on the M4 dataset when writing this. In Makridakis et al. (2020), the M4 competition for forecasting of econometric time series is introduced, as well as its winning entries. The winning entry, Smyl (2020), used a hybrid exponential smoothing and global RNN model for its forecasts, possibly inspiring some of the subsequent research on global models.

3. Data

In this section, we introduce the two datasets we use in our experiments. The M4 dataset is used in the pure forecasting experiment, and FRED is used in the zero-shot and extended dataset experiments.

The M4 dataset has time series from six domains and of six frequencies, shown in Table 1. Monthly time series dominate the dataset, and together with the yearly and quarterly frequencies

Table 1: The distribution into frequencies and domains in the M4 dataset (Makridakis et al., 2020)

Freq	Micro	Industry	Macro	Finance	Dmgr	Other	Total	Pct
Yrly	6,538	3,716	3,903	6,519	1,088	1,236	23,000	23.0%
Qrtrly	6,020	4,637	5,315	5,305	1,858	865	24,000	24.0%
Mthly	10,975	10,017	10,016	10,987	5,728	277	48,000	48.0%
Wkly	112	6	41	164	24	12	359	0.4%
Dly	1,476	422	127	1,559	10	633	4,227	4.3%
Hrly	0	0	0	0	0	414	414	0.4%
Total	25,121	18,798	19,402	24,534	8,708	3,437	100,000	100%

Table 2: The distribution over frequencies for the FRED dataset.

Freq	Yearly	Quarterly	Monthly	Weekly	Daily	Total
Num	197,920	51,913	100,808	1,728	43	352,412
Pct	56.2%	14.7%	28.6%	0.5%	0.0%	100.0%

add up to 95% of the time series. The bias towards longer frequencies reflects the focus on business applications in the M4 competition.

The Federal Reserve of Economic Data (FRED) is a database maintained by the Federal Reserve Bank of St. Louis. It contains several hundred thousand real time series from a range of domains. Table 2 shows the frequency distribution of the dataset after preprocessing. Yearly time series are heavily over-represented, and we have almost no time series of weekly, daily, and hourly frequencies.

We want a dataset with a general time series distribution as the source dataset when we pre-train models. As the M4 dataset is the downstream validation task, we interpret a general dataset to have the M4 distribution even though Fry & Brundage (2020) notes that the number of noisy and high-frequency time series is small compared to many applications. Makridakis et al. (2020) create the M4 dataset as a random subset of a larger dataset, ForeDeCk (Spiliotis, Patikos, Assimakopoulos & Kouloumos, 2021), with some constraints. Optimally we would use ForeDeCk as the source dataset but this was not available at the time of our experiments [1]. ForeDeCk

[1]ForeDeCk is now again available, but we did not find it computationally reasonable to redo all experiments with ForeDeCk as the source.

has FRED as one of its sources, and we therefore chose FRED as a source even though the data distribution is not the same as for the downstream task.

As M4 inherits from ForeDeCk which again inherits from FRED, there is a possibility of duplicated time series. Duplicates are a concern in the zero-shot experiments. Oreshkin, Carpov, Chapados & Bengio (2020a) search for duplicates in their version of FRED and the M4 dataset. In total, they find 132 duplicates, which is very small compared to the total number of time series, and therefore argue that the effect can be disregarded.

4. Modeling

This section describes the modeling approach in our experiments. The first part is a motivation for the use of global models, and the second is a specification of the models used in the experiments.

4.1. Global and Local Models

Given a dataset like the M4 dataset with multiple time series, we can differentiate between local and global modeling approaches. Local models are fitted to single time series, and we require as many models as there are time series in the dataset. Global models are fitted jointly to several time series, and we use the same model to forecast several time series. We will differentiate between global models which are fitted on the full dataset and almost global models which are fitted to the time series of a frequency.

Global and almost global models are motivated by the idea that time series share common characteristics. Accurately estimating such characteristics can be challenging if the time series are short. The M4 dataset was created with business applications in mind, and the bias towards shorter yearly, quarterly, and monthly time series reflects this. Global models can allow for more robust estimation of trend and seasonality effects for these kinds of time series, and reduce the risk of overfitting.

Learning theory aims to formalize learning and model generalization. We are interested in how ways of modeling will generalize from the training to the test set. Montero-Manso & Hyndman (2021) applies this framework to local and global time series models. We can let $|\mathcal{H}|$ denote the complexity of the model, N the number of observations for each time series, K the number of time

series, and E_{in} and E_{out} be the in-sample and out-of-sample error. Then, with probability $1 - \delta$,

$$E_{out} \leq E_{in} + \sqrt{\frac{\ln(|\mathcal{H}|) + \ln(\frac{2}{\delta})}{2NK}}. \tag{1}$$

The complexity of a local model on the whole dataset is $\prod_i^K |\mathcal{H}_L|$ as we have one model per time series. A global model will on the other hand have complexity $|\mathcal{H}_G|$. Global models can therefore be more complex than the individual local models and keep sufficient complexity for modeling the dataset. It is worth noting that the bound given above is loose, yet empirically, such bounds are often proportionally right (Abu-Mostafa, Magdon-Ismail & Lin, 2012). We refer to Vidyasagar (2003) for an introduction to learning theory and applications to real-valued functions and neural networks.

4.2. Model Specification

We use simple multilayer perceptrons (MLPs) in our experiments as we want to explore how well such simple baselines perform, and as they allow us to easily change the complexity of the model. MLPs consist of an input layer, hidden layers, and an output layer. The hidden layers have units with an activation function and weights to the next layer. For a time series model the input layer is the length of the lookback of the model, and the output is as long as the forecast. We use one-step models, meaning that we forecast only the next value. By appending forecasted values to the input we can forecast arbitrary horizons. Almost global and local models can be multi-step if the forecasting horizon is given, but we let the almost global models be one-step for better comparison to the global ones.

Large deep learning models in other domains benefit from residual and skip connections, allowing for shortcuts in the network. Residual connections add the output from one layer to a layer later in the network. Skip connections work similarly and add the residual to the last layer of the network. We allow the models to have residual and skip connections.

We know the frequency of all the time series in the dataset and let the global model access this information by appending a one-hot encoding of the frequency to the input. We believe this is fair as almost global models know which frequency they forecast, and it is a reasonable assumption that the frequency of the data is known a priori.

For better comparison to methods like N-BEATS we use ensembles of models. Our ensemble members use one of the two loss functions used to calculate the overall score in the M4 competition,

mean absolute scaled error (MASE) and symmetric mean absolute percentage error (sMAPE). For each loss function we use six different input lengths. The output of the ensemble is the median of the forecasts of the ensemble members.

In initial experiments, we found that initialization of the forecast was essential to train global MLPs. The winner of the M4 competition, Smyl (2020), use a similar approach where the residuals from a model are modeled using a recurrent neural network. We initialize the forecast with a naive or seasonal naive forecast.

5. Experiments, Results, and Discussion

In this section, we describe and discuss the results of our three experiments. We first use the M4 dataset as one would have done as a participant in the M4 competition. Next, we use the FRED dataset as a source dataset to train models and then use these to forecast the M4 dataset. Finally, we extend the M4 dataset with proportions of the FRED dataset. We report the performance of the models as in the M4 competition, using overall weighted average (OWA). This is a scaled metric against the performance of a naive forecaster adjusted for seasonality on the dataset. Formally, the OWA score is defined as,

$$\text{OWA} = \frac{1}{2}\frac{\text{MASE}}{\text{MASE}_{\text{NAIVE2}}} + \frac{1}{2}\frac{\text{sMAPE}}{\text{sMAPE}_{\text{NAIVE2}}}, \tag{2}$$

where MASE is the mean average scaled error, sMAPE is the symmetric mean absolute percentage error, and NAIVE2 is a seasonally adjusted naive forecaster (Makridakis et al., 2020).

5.1. Global Models on the M4 Dataset

In this experiment, we use the M4 dataset to train global and almost global MLPs. We create a validation dataset with the same size as the test dataset using the end of the training dataset. Appendix C describes the model selection process on the validation set. We refer to Appendix B for specific implementation details outside of what is described here.

The global model is a 10 layer MLP with 1024 hidden units in each layer, ReLU activation functions, and a dropout rate of 0.2. We use a naive initialization of the forecast using the previous value. We one-hot encode the frequencies and append these to the input. The full method is an ensemble using the two loss functions described earlier, and input lengths $[20, 30, 40, 50, 60, 70]$. In total, the method has 12 models with around 8.5M parameters each.

The almost global model is similar, but also use residual and skip connections for every fourth layer. We use one model for each frequency, and create an ensemble in a similar way with two loss functions and 6 input lengths. The input lengths of the models are $[2, 3, 4, 5, 6, 7]$ times the forecasting horizon as in N-BEATS. In total, this method has 72 models.

Table 3 shows the results of the global and almost global model on the test, the best ensemble members, top entries from the M4 competition, and benchmarks from the competition. The almost global model without is the best model in terms of overall score, scoring similar to the best entry from the M4 competition, ES-RNN (Smyl, 2020). The global model is not far behind and models all frequencies well except for the hourly segment.

Compared to the methods from the M4 competition the global and almost global models perform well. Both beat all benchmark methods and score better than the 2^{nd} best method from the competition, FFORMA (Montero-Manso, Athanasopoulos, Hyndman & Talagala, 2020). Most interesting is the difference from the local MLP benchmark from the competition which placed 57^{th} of the 61 methods. The currently best performing method is N-BEATS, which is a multi-step almost global method with a specific architecture and a larger ensemble.

The best ensemble members perform well and are close to the full ensemble. Based on the performance on the validation set we selected models with sMAPE loss and input length 70 for the global model, and MASE loss and input length 2 times the forecasting length for the almost global model. We selected the best members based on one run, and the randomness from the initialization of the network can therefore cause us to select the wrong model. On the test set a global model with sMAPE loss and input length 40 scores 0.832, and an almost global model with MASE loss and input length 3 scores 0.828. Performing multiple runs could smooth out some of the randomness.

Figure 1 shows the performance of the ensemble members of N-BEATS, the global, and the almost global model. Ensemble members in N-BEATS are weaker but work better together. This observation indicates that one of the main strengths of N-BEATS is that it is able to create members that work well together. This may be a consequence of both the training pipeline and the N-BEATS architecture.

Table 3: Overall scores of models on the M4 test set. Numbers in parentheses indicate the ranking from the M4 competition. The best single models are models that perform best on the validation set.

Model	Yearly	Qrtrly	Monthly	Weekly	Daily	Hourly	OWA
Global	0.772	0.841	0.857	0.936	0.954	1.086	0.825
Global single	0.796	0.851	0.886	0.923	0.940	1.531	0.851
Almost global	0.771	0.840	0.852	0.719	0.980	0.577	0.821
Almost global single	0.787	0.855	0.851	0.901	0.990	0.728	0.830
N-BEATS	0.765	0.800	0.820	0.822[2]	-	-	0.797
ES-RNN (1)	0.778	0.847	0.836	0.851	1.046	0.440	0.821
FFORMA (2)	0.799	0.847	0.858	0.796	1.019	0.484	0.838
Theta (18)	0.872	0.917	0.907	0.971	0.999	1.006	0.897
Comb (19)	0.867	0.890	0.920	0.926	0.978	1.556	0.898
ARIMA (29)	-	-	-	-	-	-	0.903
MLP (59)	1.288	1.684	1.749	3.608	3.509	0.921	1.642

5.2. Zero-Shot Forecasting of the M4 Dataset Using the FRED Dataset

In our next experiment, we forecast the M4 dataset using models trained on proportions of the FRED dataset. Each smaller training proportion is a subset of the larger proportions. We enforce that the proportion is the same for each frequency when we sample the proportion from FRED. As the FRED dataset does not have an hourly segment we omit this frequency for the almost global model. The global model can still handle unseen frequencies and we therefore keep hourly series in the test set for these models. We reuse the models and learning schedules from the previous experiment.

Since the FRED dataset is larger than the M4 dataset it might allow for more complex models. In addition to the model from the first experiment we train a smaller model with 5 layers and 512 hidden units in each layer, and a larger model with 10 layers and 2048 hidden units in each layer. We only train the smaller and larger models for the global model due to the large number of models required for the almost global model. For the same computational reasons, we restrict the experiments to ensembles with one loss function, sMAPE.

Figure 2a shows the score of the scores of the global and almost global models trained on

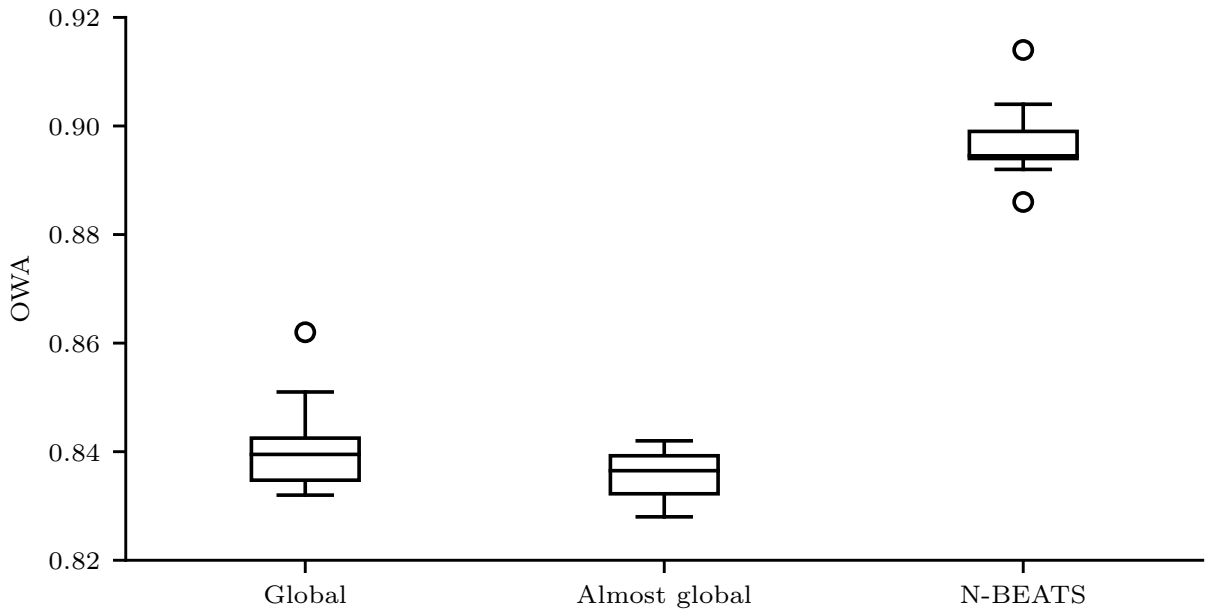
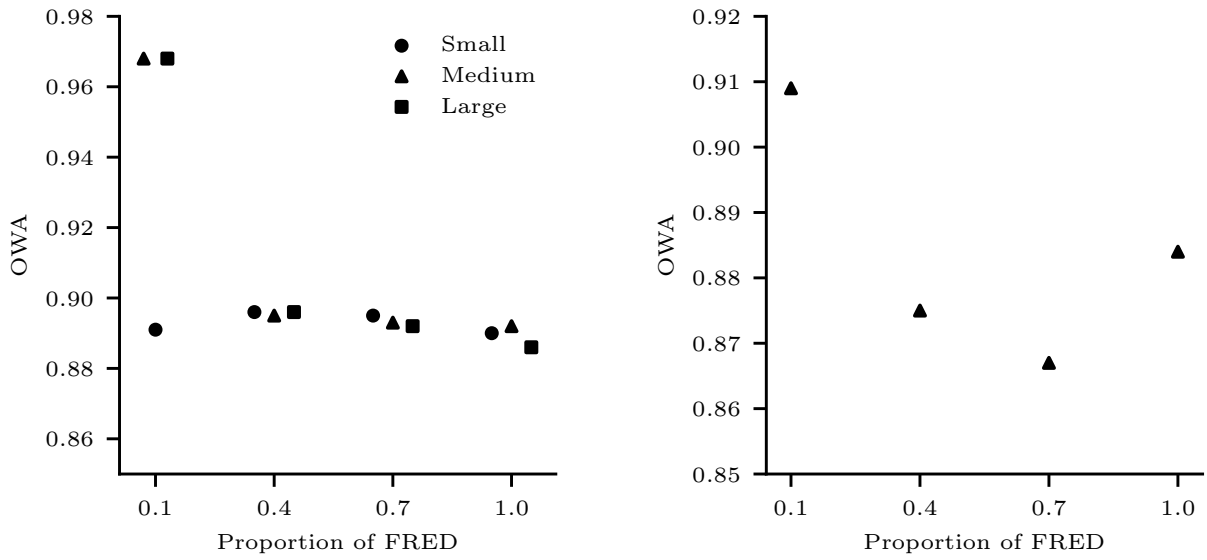


Figure 1: The OWA scores of the ensemble members of the global MLP, almost global MLP, and N-BEATS.

proportions of the FRED dataset and tested on the M4 dataset. The models perform better with more data even though added performance using more than 40% of the dataset is limited. With limited data, 10%, the larger models struggle to learn. The small model performs well already here, supporting better generalization with less data for less complex models. A relatively small global MLP with 5 layers and 512 hidden units trained on only 10% of the FRED dataset outperforms the benchmarks in the M4 competition without using any of the M4 training data.

The almost global models in Figure 2b outperform the global models. There is an increase in performance until we reach 70% of the dataset. Up until this point, the almost global model see a more clear increase in performance with more data. We do not see why the last 30% would lead to worse performance, but with only one run of the experiment we can be unlucky with the last 30% containing confusing time series for the model.

The strong performance of both global and almost global models in this zero-shot experiment is a strong indicator that common characteristics of time series are well estimated using global models.



(a) Global

(b) Almost Global

Figure 2: Zero-shot total OWA scores for global and almost global models. The scatter points for the global models have been slightly shifted along the x-axis for better readability.

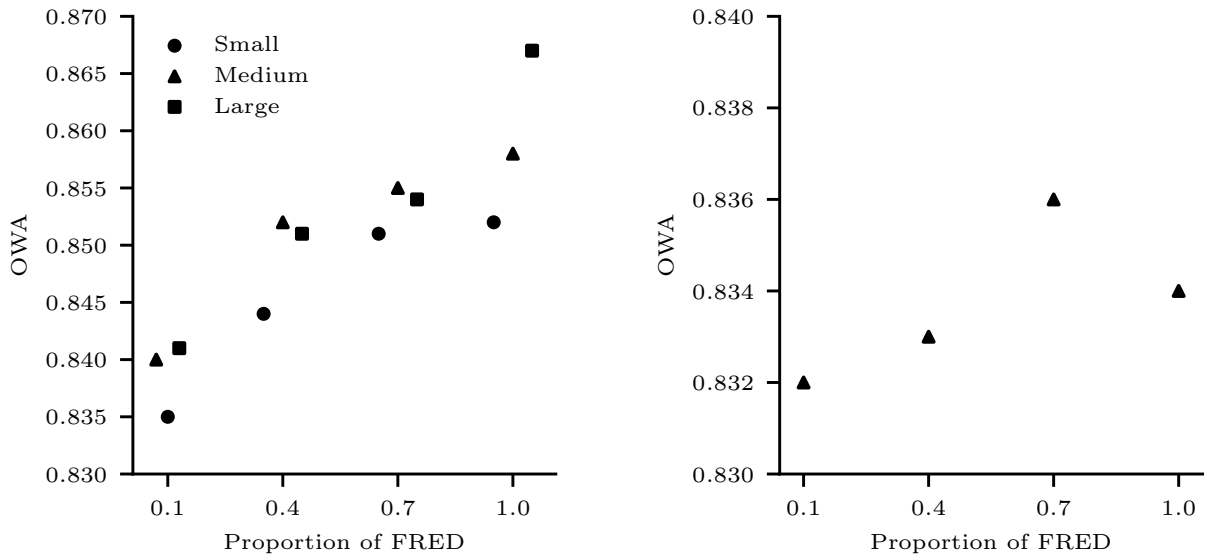
5.3. Extending the M4 Dataset With the FRED Dataset

A continuation of the previous two experiments is to use both the M4 and FRED datasets together. In this experiment, we use the M4 dataset and add proportions of the FRED dataset to the training dataset. An extended dataset can potentially allow for more complex models with less risk of overfitting. We use a similar setup as in the zero-shot experiment, but include M4 in the training dataset in addition to the proportions of FRED.

Figure 3a shows the OWA score of the three global models. All models perform better using as little as the FRED dataset as possible. The smaller model is also better for all proportions.

The increased OWA score as we add more of the FRED dataset is an indicator that the M4 dataset is sufficiently large for the models to learn and generalize well. Given that there is enough data to model the distribution well, more data from a slightly different distribution can move the model away from the target distribution. The small global model is least affected by the added data. A less complex model is less affected by more data as its capacity restricts it from utilizing all the additional data from FRED.

The scores of the almost global models are shown in Figure 3b. While these models also perform



(a) Global

(b) Almost global

Figure 3: Overall scores for global and almost global models trained on M4 and proportions of FRED. The models and setup is as in Figure 2.

better with as little data from FRED as possible, the scale of the change is much smaller.

Global and almost global models behave differently with an extended dataset. The changes in the global models are on a larger scale than the almost global models. A possible contributor to this effect is that the global model will have to model all frequencies together. The FRED dataset is biased towards yearly time series, and the M4 dataset is biased towards monthly time series. When we calculate the loss we will see more and more yearly time series in the global model. Optimizing for the loss will therefore turn towards optimizing for yearly time series. Almost global models behave differently as the optimization is done for each frequency separately. These models are therefore not prone to changes in the distribution over frequencies, but will not be affected by changes within each distribution.

6. Conclusion

In this paper, we have demonstrated how global MLPs can be trained to forecast the M4 dataset well. Global models learn common characteristics of time series and can therefore also be used successfully for zero-shot forecasting.

Global and almost global MLPs perform on par with the best entry from the M4 competition, contrary to the local MLP benchmark which placed third to last. They learn common characteristics for time series as a model using the FRED dataset as source outperformed all benchmarks from the M4 competition without being fitted to the training data. However, adding more data to the M4 dataset did not improve any of the models, likely because the M4 dataset is already large enough. In cases with limited data, we could potentially benefit from sourcing additional data or using pre-trained models for transfer learning. The best single models in our ensemble experiments are close in performance to the full ensemble while having far fewer parameters and FLOP requirements than the entire ensemble. This makes them desirable for real-world applications, where the cost of inference needs to be taken into consideration. While global deep learning models perform well, the steps to make them work well are not entirely straightforward.

The success of global models on the M4 dataset motivates future work on global models. All benchmarks in the M4 competition were local models, but recent work on global models suggests that there should be global benchmark models in future competitions.

References

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012). *Learning from data* volume 4. AMLBook New York, NY, USA.

Federal Reserve Bank of St. Louis (2021). FRED - economic data since 1991. URL: `https://fred.stlouisfed.org/`.

Fry, C., & Brundage, M. (2020). The m4 forecasting competition – a practitioner's view. *International Journal of Forecasting*, *36*, 156–160.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., & ravis E. Oliphant (2020). Array programming with NumPy. *Nature*, *585*, 357–362. doi:`10.1038/s41586-020-2649-2`.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *International Conference on Learning Representations*, .

Makridakis, S., Spiliotis, E., & Assimakopoulos, V. (2020). The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, *36*, 54–74.

Montero-Manso, P., Athanasopoulos, G., Hyndman, R. J., & Talagala, T. S. (2020). Fforma: Feature-based forecast model averaging. *International Journal of Forecasting*, *36*, 86–92.

Montero-Manso, P., & Hyndman, R. J. (2021). Principles and algorithms for forecasting groups of time series: Locality and globality. *International Journal of Forecasting*, .

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020a). Meta-learning framework with applications to zero-shot time-series forecasting. *arXiv preprint arXiv:2002.02887*, .

Oreshkin, B. N., Carpov, D., Chapados, N., & Bengio, Y. (2020b). N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=r1ecqn4YwB.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradamadbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., & Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 32* (pp. 8024–8035). Curran Associates, Inc. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

Själander, M., Jahre, M., Tufte, G., & Reissmann, N. (2019). EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure. *arXiv preprint arXiv:1912.05848*, .

Smyl, S. (2020). A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, *36*, 75–85.

Spiliotis, E., Patikos, A., Assimakopoulos, V., & Kouloumos, S. A. (2021). FOREDeCk: The FOREcasting & Strategy Unit Data Collection. URL: http://fsudataset.com/.

pandas development team, T. (2020). pandas-dev/pandas: Pandas. doi:10.5281/zenodo.3509134.

Vidyasagar, M. (2003). Vapnik-chervonenkis, pseudo-and fat-shattering dimensions. In *Learning and Generalisation* (pp. 115–147). Springer.

Williams, Z. (2015). Python wrapper of the st. louis federal reserve bank's fred api web service for retrieving economic data. https://github.com/zachwill/fred.

Appendix A. Data

This appendix describes how we sourced the FRED dataset and includes a discussion about datasets for global models.

Sourcing the FRED Dataset

The FRED dataset consists of all time series from the Federal Reserve of Economic Data Federal Reserve Bank of St. Louis (2021). There is an API allowing interaction with the database, but no straightforward way to download the whole database and no reference to the structure of the database. We use a python wrapper around the lower-level API to interact with the database Williams (2015). A personal API key is required and can be obtained through an online form.

Each time series has links to its parent and children allowing us to traverse the dataset to access all time series. After collecting all the time series IDs we can download the time series. Downloading the dataset takes between 5-8 days due to the restriction on the frequency of API calls.

Preprocesing the FRED Dataset

The first step of our preprocessing is to require a minimum length of the time series. We use the same length constraints as was done when the M4 dataset was created. The minimal lengths are 19 for yearly, 24 for quarterly, 60 for monthly, 93 for weakly, and 107 for daily.

Next, we try to save time series with missing values. We keep sub-sequences without missing values that pass the length constraints. By doing this we save long time series with only a few missing values.

Some time series are constant or only have a few distinct values. In theory, this should be learnable by the models, but we found that this in some cases introduced noise. We drop time series with less than five distinct values. In line with this, we also drop time series with more than 50% zeros. Lastly, we enforce the positivity constraint from the M4 competition and therefore drop the time series with negative values.

These preprocessing steps reduce the number of time series from the 686,623 downloaded time series to the 352,412 time series shown in Table 2.

Temporal bias for Datasets when using Global Models

Global models pose a challenge of data snooping when we split time series into training and test sets using the last observed horizon as in the M4 competition. If we have two related time series ending at different dates, the training split of one will continue into the test split of the other. A global model can therefore learn useful characteristics from the training split of the first that can be useful in forecasting the test split of the second.

Most entries and all benchmarks in the M4 competition were purely local models. Temporal bias caused by the data split can not be utilized by these models. The M4 split is the typical way to split the data, but other ways can remove the temporal bias from global models. However, these have their separate flaws.

First, we can split by date rather than using the last part of each horizon. This approach is possible when we have metadata about the observations. However, we will disregard much of the data if we also require the time series to have the same forecasting length, and if we allow for arbitrary forecasting lengths we restrict ourselves to one-step models. In addition, we introduce a bias towards newer time series in the test set. Most forecasting applications car mostly about the future. Still, in for example economics, we want to see the model under different economic regimes.

Second, we can disregard all discontinued time series. Under this approach, splitting at a date and splitting at the last horizon coincides. One drawback is that we will potentially throw away useful data as time series databases like FRED contain historic data. More severe is probably the introduction of survivor bias. The distribution we model will then be biased towards companies, countries, and metrics that survive, which can deviate from the distribution in our application.

Appendix B. Implementation details

This appendix describes the implementation details for our experiments. All code is available in a GitHub repository, `https://github.com/erikdengerud/Global-TS-Forecasting`.

Software and Hardware

Deep learning models are implemented using PyTorch (Paszke, Gross, Massa, Lerer, Bradamadbury, Chanan, Killeen, Lin, Gimelshein, Antiga, Desmaison, Kopf, Yang, DeVito, Raison, Tejani, Chilamkurthy, Steiner, Fang, Bai & Chintala, 2019), with help from NumPy (Harris, Millman, van der Walt, Gommers, Virtanen, Cournapeau, Wieser, Taylor, Berg, Smith, Kern, Pi-

Dataset	Metric	Yrly	Qrtrly	Mnthly	Wkly	Dly	Hrly	Total
Val	MASE	4.723	1.378	1.063	2.898	3.221	11.532	2.109
	sMAPE	19.251	11.247	13.102	9.196	2.778	41.399	13.738
Test	MASE	3.974	1.371	1.063	2.777	3.278	2.396	1.912
	sMAPE	16.342	11.012	14.427	9.161	3.045	18.383	13.564

Table B.4: Naive2 scores used to scale the OWA score for the validation and test sets.

cus, Hoyer, van Kerkwijk, Brett, Haldane, del R'ıo, Wiebe, Peterson, G'erard-Marchant, Sheppard, Reddy, Weckesser, Abbasi, Gohlke & ravis E. Oliphant, 2020) and Pandas (pandas development team, 2020). Comparisons with the ensemble members of N-BEATS are done using the authors' code available on GitHub, `https://github.com/ElementAI/N-BEATS`. Code used to calculate Naive2 scores for the validation set is available from the official M4 repository at `https://github.com/Mcompetitions/M4-methods`. The models are trained using the NTNU Idun computing cluster Själander, Jahre, Tufte & Reissmann (2019).

Data

We use the M4 and FRED datasets in our experiments. For the M4 dataset we create a validation set using the last forecasting horizon of the training set. The validation set is used for model selection. Calculated Naive2 scores for the validation set and Naive2 scores from the M4 competition are shown in Table B.4. A discussion of data for global models and the sourcing and preprocessing of the FRED dataset can be found in Appendix A.

Sampling During Training

We sample training samples according to Algorithm 1. During training we use a batch size of 1024. The algorithm is similar to the one used in N-BEATS, with the main difference that we sample for one-step methods. Each epoch has one sample from each time series, but since we sample random cut points the samples will not be the same in each epoch. The number of training samples the model sees is therefore larger than the number of time series in the dataset. As a way to stabilize training, we sample cut points from the end of the times series as is also done in N-BEATS. Doing this reuses more of the most relevant parts of the time series and also makes sure the training samples are not too short. We sample cut points among the last n observations, where

Algorithm 1: Sampling algorithm

Result: One epoch of training samples and labels.

require a $c_{frequency}$ for each frequency, determining how close to the end of the training set we sample from.;

require the length of the input window of the network.;

for *each time series in the dataset* **do**

  Sample a cut point in the last $c_{frequency}$ points of the time series. ;

  Use the cut point as a label and the observations before it as the sample. ;

  if *the length of the sample is greater than the input window* **then**

    Cut from the beginning of the sample so that it has the same length as the input window.

  else

    Zero pad the beginning of the sample so that it has the same length as the input window.

  end

end

n is 6 for yearly, 8 for quarterly, 18 for monthly, 13 for weekly, 14 for daily, and 48 for hourly data. For models that use scaling we scale by dividing the whole sample by the max value of the sample.

Modeling Residuals

We are not able to train models successfully unless we model residuals rather than direct values. In the implementation, we model the residuals by adding the last observed value to the output of the network. This makes the model learn how to predict the change from the previous value. We can also initialize the forecast with a naive seasonal forecast by adding the observation from the previous season to the output.

Optimization

We use the Adam optimizer Kingma & Ba (2014) with default parameters. We optimize for total sMAPE or MASE in the experiments. This differs from optimizing the scaled OWA which is the final test score.

During training we use a learning rate schedule to adjust the learning rate. We reduce the learning rate by a factor of ten when we hit a plateau on the validation set. The model then

has a number of epochs at the new learning rate to find a better model. This number is called patience and we use patience of 10 epochs in the model selection phase. We stop training the models whenever we have reduced the learning rate to its minimum value or the model has not improved in 30 epochs. This number is often called tenacity and we use high patience and tenacity due to the randomness of the sampling algorithm.

During testing we use a slightly different scheme where we half the learning rate every 20 epochs for 200 epochs. We found this scheme to perform similarly on the validation set but easier to implement when we do not have a validation set at test time.

To prevent overfitting we use dropout with a value of 0.2. As some deep learning models have trouble with large gradients we also clip gradients at 1.

Ensembling

A key to the success of N-BEATS is ensembling weaker models. We use ensemble members with the same input lengths as N-BEATS for the almost global model. The members have input lengths $[2, 3, 4, 5, 6, 7]$ times the forecasting horizon of the frequency, and loss functions sMAPE and MASE.

We have to change this slightly for the global models as we have different forecasting horizons for each frequency. Initially, we choose an ensemble with input lengths that cover most of the input lengths of the members of the almost global ensemble, $[30, 60, 90, 120, 150, 180]$.

Performance Metrics

We use the same performance metrics as in the M4 competition. These are the Mean Absolute Scaled Error and Symmetric Mean Absolute Percentage Error,

$$\text{MASE} = \frac{1}{h} \frac{\sum_{t=n+1}^{n+h} |y_t - \hat{y}_t|}{\frac{1}{n-m}\sum_{t=m+1}^{n} |y_t - y_{t-m}|}, \tag{B.1}$$

$$\text{sMAPE} = \frac{2}{h} \sum_{t=n+1}^{n+h} \frac{|y_t - \hat{y}_t|}{|y_t| + |\hat{y}_t|} \times 100\%. \tag{B.2}$$

Here, h is the forecasting horizon, n is the number of observations, and m is the seasonality of the time series. MASE is scaled intuitively against the error of a naive seasonal forecast. The period for the frequencies are defined in Table B.5.

Frequency	Yrly	Qrtrly	Mnthly	Wkly	Dly	Hrly
Seasonality	1	4	12	1	1	24

Table B.5: Seasonalities used in MASE in the M4 competition.

The overall score in the M4 competition is a weighted average of MASE and sMAPE scaled by how well a naive seasonal forecast adjusted for trend (NAIVE2) performs on the dataset. Overall Weighted Average score (OWA) is defined by,

$$\text{OWA} = \frac{1}{2}\frac{\text{MASE}}{\text{MASE}_{\text{NAIVE2}}} + \frac{1}{2}\frac{\text{sMAPE}}{\text{sMAPE}_{\text{NAIVE2}}}, \tag{B.3}$$

and is the metric we use to compare models.

Appendix C. Model Selection

This appendix is a walk-through of the steps in the model selection process. We imitate the process one would follow in the M4 competition by creating a train and validation set from the original M4 train set. The last horizon for each frequency is used as the validation set. We calculate the Naive2 scores on the validation set to get correct OWA scores for the validation set. The Naive2 calculation is done using a modified version of the benchmark code from the M4 GitHub repository [3].

The rest of the appendix is divided into two sections, one for almost global and one for global models. Both sections have the same structure, selecting depth and width, finding an appropriate learning schedule, deciding on the type of initialization of the forecast, determining the effect of residual and skip connections, and lastly, whether inputs should be scaled. In addition, the global selection process includes experiments on the input lengths of the ensemble members.

Almost Global Model Selection

The baseline model in the experiments has 10 layers with 512 hidden units. Skip and residual connections are added every fourth layer. We use ensemble members with sMAPE loss functions and input lengths $[2, 3, 4, 5, 6, 7]$ times the forecasting horizon. The input lengths are the same as in N-BEATS. Initially, we do not scale the inputs to the model, and we use a naive initialization

[3] `https://github.com/Mcompetitions/M4-methods`

Table C.6: Overall score for the almost global model on the validation set as a function of the number of hidden units in each layer.

Hidden units	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
64	0.733	0.927	0.908	0.890	**1.107**	0.256	0.839
128	0.728	0.921	0.897	0.836	1.109	0.252	0.832
256	0.726	0.916	0.888	0.820	1.110	0.241	0.827
512	**0.720**	**0.909**	0.882	0.804	1.113	0.236	0.822
1024	0.721	**0.909**	**0.879**	0.790	1.114	0.231	**0.821**
2048	0.724	0.912	0.881	**0.766**	1.115	**0.217**	0.823

of the forecasts. The architecture experiments were done using a plateau learning schedule where the model changes its learning rate if the performance on the validation set plateaus. This is not sound in the sense that the validation set is used to score the models and the models have snooped at the validation set. Later we end up with a different schedule and we do not believe the selected architecture would be much different with the chosen schedule.

Model Architecture

Table C.6 shows the almost global model's overall score as a function of the number of hidden units in each layer. The model with 1024 hidden units performs best. It is closely followed by the 512 and 2048 models, but models the monthly segment slightly better. Wider models seem to model especially the shorter frequencies better.

The overall scores as we vary the depth of the network are shown in Table C.7. Models with between 5 and 20 layers all perform well, with the 5 and 10 layer models standing out. Deeper models struggle especially in the monthly segment, and the 50 layer model does not seem to converge properly even with residual and skip connections. Better regularization or another training schedule might be needed to get very large models to learn properly.

There is a subset of possible best models based on the width and depth experiments. These vary a lot in the total number of parameters. Table C.8 shows the overall scores of the models on the validation set. It is the largest model that performs the best. The models are all very similar in performance, but it is again better modeling of the monthly segment that puts the largest model ahead.

Table C.7: Overall score of the almost global model on the validation set as a function of the number of layers.

Layers	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
1	0.853	0.967	1.003	0.928	1.110	0.257	0.931
5	0.725	**0.907**	**0.883**	**0.757**	1.115	**0.204**	0.823
10	**0.721**	0.911	**0.883**	0.793	1.112	0.232	**0.822**
15	0.723	0.915	0.886	0.811	1.112	0.255	0.825
20	0.728	0.921	0.889	0.861	1.109	0.286	0.830
30	0.745	0.930	0.911	0.879	1.106	0.784	0.856
50	0.830	1.057	1.104	0.992	**1.102**	0.956	0.979

Table C.8: Overall score of the almost global model on the validation set for candidate architectures. W and D in the first columns indicate the width and depth of the model.

W - D	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
512 -5	0.725	**0.907**	0.883	**0.757**	1.115	0.204	0.823
512 - 10	**0.721**	0.911	0.883	0.793	**1.112**	0.232	0.822
1024 - 5	0.728	0.908	0.881	0.761	1.113	**0.200**	0.824
1024 - 10	**0.721**	0.909	**0.879**	0.790	1.114	0.231	**0.821**

Learning Rate Schedules

The scores of the learning rate schedules can be seen in Table C.9. Reducing on plateaus and the 10 step schedule is comparable with the 10 step schedule being slightly better. Also, this schedule is easily applicable without a validation set.

Initialization of Forecasts

Table C.10 shows the results for naive, seasonal naive, and no initialization.

Residual and Skip Connections

We test for the effect of the residual and skip connections in Table C.11. The model with residual and skip connections is better, but the difference is not large.

Table C.9: Overall score of the almost global model on the validation set for different learning rate schedules. We use the model with 10 layers and 1024 hidden units.

Schedule	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Plateau	0.721	**0.909**	0.879	0.790	**1.114**	0.231	0.821
3-step	0.904	0.984	1.539	3.110	2.116	0.540	1.152
10-step	**0.717**	0.913	**0.876**	**0.765**	1.143	**0.193**	**0.819**
Cosine	0.913	0.924	0.881	1.569	2.139	0.210	0.950

Table C.10: Overall scores of the almost global model on the validation set for naive, seasonal naive and no initialization. We use models with 10 layers and 1024 hidden units. All models are trained with the plateau schedule.

Initialization	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
None	60.000	376.491	6.0E3	3.1E3	8.5E3	1.8E10	4.0E8
Naive	**0.717**	0.913	**0.876**	**0.765**	1.143	0.193	**0.819**
S-naive	0.719	**0.909**	0.882	0.774	**1.112**	**0.186**	0.820

Scaling of Inputs

Table C.12 shows the effect of scaling the inputs using max-scaling. Scaling the inputs is not necessary, and the model performs better with the raw data. Keeping the raw data allows for the model to use the scale in the data. Montero-Manso & Hyndman (2021) see improvement by scaling and feeding the scale as input to the model.

Global Model Selection

We use the same baseline model as in the almost global experiments, 10 layers, and 512 hidden units. Again, we use ensemble members with sMAPE loss function. For the baseline model, we

Table C.11: Overall score of the almost global model on the validation set with and without residual and skip connections. All models are trained with the plateau schedule. The baseline model is the same run as in Table C.9. The None model is the model without any skip or residual connections.

Model	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Res and skip	**0.717**	0.913	**0.876**	**0.765**	1.143	**0.193**	**0.819**
None	0.726	**0.912**	0.887	0.834	**1.110**	0.231	0.827

Table C.12: Overall scores of the almost global model on the validation set with and without scaling of the inputs.

Model	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Unscaled	**0.717**	0.913	**0.876**	**0.765**	1.143	0.193	**0.819**
Scaled	0.757	**0.898**	0.879	0.845	**1.109**	**0.182**	0.834

Table C.13: Overall scores of the global model on the validation set as a function of the number of hidden units in each layer. We use the baseline model with 10 layers in the experiment and only change the number of hidden units.

Hidden units	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
64	0.730	0.985	0.969	0.909	1.152	6.431	1.005
128	0.728	0.988	0.973	0.912	1.148	5.017	0.974
256	0.726	0.963	0.929	0.889	1.162	**2.928**	0.908
512	**0.725**	0.955	0.918	0.890	1.157	3.273	0.910
1024	0.727	0.949	**0.913**	**0.888**	**1.136**	3.098	**0.902**
2048	0.730	**0.945**	0.922	0.891	1.137	4.686	0.942

choose input lengths $[30, 60, 90, 120, 150, 180]$ as this is in the range of many of the almost global models. Finding appropriate input lengths is an additional experiment in this section.

Model Architecture

Table C.13 show the overall score for each frequency and in total. The model with 1024 hidden units is the best performer. Interestingly, 256 units also perform well, and this is a model with a lot fewer parameters. We see the most variety in performance in the hourly segment. The hourly segment is small and is a minor contributor to the total score. However, when the overall score for the hourly segments gets very large, it will be a factor. All models struggle to forecast the hourly series accurately. As the same model is used for all time series, the poor performance likely is because the hourly series are different from the rest. Since the proportion of hourly series is so small, it will make sense for the network to learn how to predict the other frequencies.

The performance as we vary the depth of the network is seen in Table C.14. While the model with 5 layers performs best in most segments, it is the model with 10 layers that has the best overall score. Again it is the hourly segment that varies the most and is hard to forecast. The hourly segment is what prevents the 5 layer model from being the best model. Interestingly, the model

Table C.14: Overall scores of the global model on the validation set as a function of the number of layers. We use the baseline model with 512 hidden units in the experiment and only change the number of layers. The model with 10 layers is equivalent to the model with 512 hidden units in Table C.13, but with different random initializations.

Layers	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
1	0.885	1.034	1.145	1.071	1.229	3.729	1.081
5	0.728	**0.940**	**0.908**	**0.874**	1.141	5.542	0.955
10	**0.723**	0.951	0.919	0.890	1.143	3.405	**0.911**
15	0.724	0.961	0.924	0.898	1.138	4.467	0.939
20	0.728	0.981	0.952	0.899	1.146	2.714	0.915
30	0.736	1.045	1.070	0.961	1.147	1.936	0.949
50	1.045	1.073	1.105	0.999	**1.104**	**0.939**	1.073

with 50 layers is the only one that is close to forecast hourly data. However, poor performance on all other frequencies prevents the model from being competitive.

We continue with a model with 10 layers and 1024 hidden units in each layer based on our experiments. However, we are not far from using a model with 5 layers and 256 hidden units. The first model is almost 40 times larger than the second measured in the number of parameters. If we believe we are risking overfitting, we could shift to the smaller model.

Input Lengths of Ensemble Members

Figure C.4 shows the overall score of ensemble members as a function of input window length. We test input window lengths from the shortest local models to the longest. The performance seems to stabilize from input length 70. For the new ensemble, we choose members evenly distributed in the area that perform well, $[70, 110, 150, 190, 230, 270]$.

Learning Schedules

Table C.15 shows the total overall scores for the three learning rate schedules on the validation set together with the previously used plateau schedule. None of the schedules are better than reducing on a plateau. Still, the aggressive step-wise schedule is comparable in the most important segments, is easier to implement on a test set, and does not have any implicit information about the validation set. Cosine annealing is not modeling the hourly segment well, and the three-step

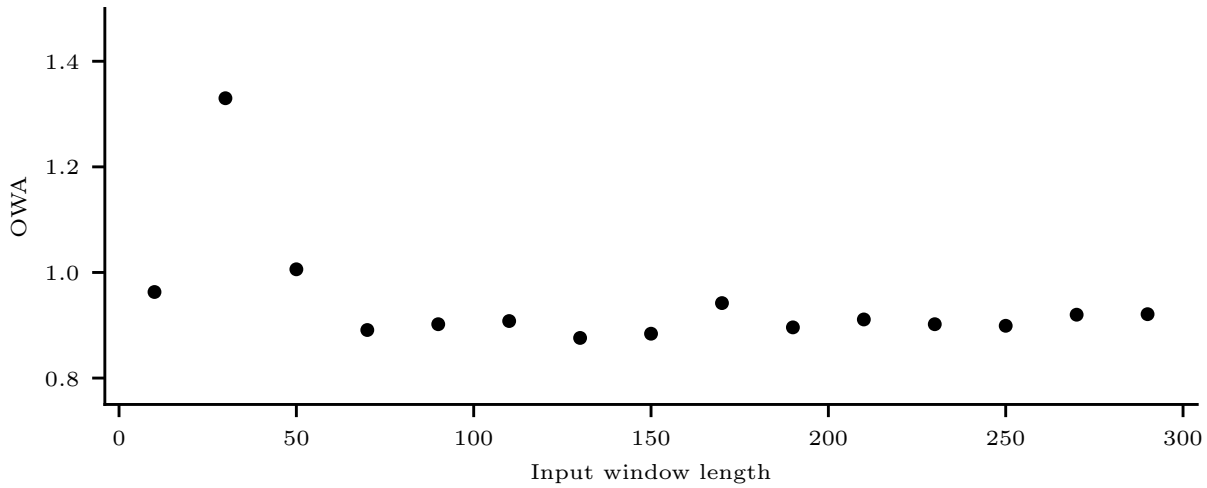


Figure C.4: Overall score of candidate ensemble members as a function of input window length for a model of 10 layers with 1024 hidden units.

Table C.15: Overall scores of the global model on the validation set for 3-step, 10-step and cosine learning schedules. We use models with 10 layers and 1024 hidden units.

Initialization	Yearly	Qrtrly	Monthly	Weekly	Daily	Hourly	OWA
Plateau	**0.727**	0.949	0.913	**0.888**	1.136	3.098	**0.902**
Cosine	0.860	0.968	**0.891**	0.909	**1.093**	149.540	1.675
3-step	0.797	1.346	2.802	3.186	6.093	**2.828**	1.746
10-step	0.728	**0.943**	0.916	0.897	1.129	4.650	0.937

schedule is not modeling any frequencies well except the hourly. We ran this experiment using the original ensemble.

Initialization of Forecasts

Table C.16 shows the results with naive, seasonal naive, and no initialization of the forecast. We see that we need initialization, but contrary to the almost global model we need naive initialization.

Encoding Frequencies

Seasonal initialization is one way to incorporate information about the frequency. Another is to encode the frequency and concatenate this with the input. We encode frequencies as one-hot encoded vectors, allowing the model to more explicitly use frequency information.

Table C.16: Overall scores of the global model on the validation set for naive, seasonal naive and no initialization. We use models with 10 layers and 1024 hidden units. All models are trained with the plateau schedule.

Initialization	Yearly	Qrtrly	Monthly	Weekly	Daily	Hourly	OWA
None	67.291	349.858	2.0E5	1.6E4	3.5E4	2.0E14	1.0E12
Naive	**0.727**	**0.953**	**0.922**	**0.907**	**1.143**	2.629	**0.897**
S-naive	1.256	1.511	1.183	45.056	60.982	**2.436**	4.664

Table C.17: Overall score of the global model on the validation set with and without encoding the frequency. All models are trained with the plateau schedule.

Model	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Base	0.727	0.953	**0.922**	0.907	**1.143**	2.629	0.897
Encoding	**0.726**	**0.950**	0.924	**0.901**	1.144	**1.704**	**0.876**

Table C.17 shows the model with and without encoding the frequency. The total overall score improves, mainly due to better modeling of the hourly segment. So far the hourly frequency has been the most challenging to model for global models. In all other segments, the baseline model and the model with encoded frequencies are comparable. We could have encoded the specific domain of the time series as well. However, this would have been unfair to the almost global models and made the model very specific to the dataset.

In the context of global pre-trained models, we can use the same encoding even with unknown frequencies by having an encoding for unknown frequencies.

Residual and Skip Connections

Again, we test for the effect of the residual and skip connections. Table C.18 shows the results when removing the residual and skip connections for the global model. Contrary to the almost global model without connections is better.

Scaling of Inputs and an Alternative Ensemble

Lastly, we scale the inputs to the model using max-scaling. In addition, we test an ensemble with ensemble members that have shorter input window lengths. The motivation for the alternative ensemble is that ensemble members with shorter input lengths have performed better in the last experiments. We create an alternative ensemble with input window lengths $[20, 30, 40, 50, 60, 70]$.

Table C.18: Overall score of the global model on the validation set with and without residual and skip connections. All models are trained with the plateau schedule.

Model	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Res + skip	**0.726**	0.950	0.924	0.901	1.144	1.704	0.876
None	**0.726**	**0.945**	**0.922**	**0.890**	**1.134**	**1.456**	**0.869**

Table C.19: Overall score of the base and alternative ensemble global model on the validation set with and without scaling of inputs.

Model	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly	OWA
Base	0.733	0.930	0.899	0.874	**1.131**	0.476	0.842
Base scaled	**0.725**	0.924	0.894	0.942	1.152	**0.363**	0.835
AE	0.733	0.926	0.888	**0.855**	1.150	0.611	0.840
AE scaled	0.729	**0.912**	**0.885**	0.910	**1.131**	0.421	**0.831**

Table C.19 shows the overall scores with and without scaling for the current ensemble, and the alternative ensemble. Both scaling and the alternative ensemble improves the model.

Global models benefit from scaling the inputs while almost global models do not. One reason can be that global models have to learn all frequencies together. While scaling removes information, it can allow for better learning across frequencies by making the time series more comparable.