Hierarchical Load Forecasting: 2017 Global Energy Competition - Final report

Inés Dormoy, Louis Gautier
ICME & MS&E, Stanford University
idormoy@stanford.edu, lgautier@stanford.edu
The code for our project is available on this repository

Abstract

Forecasting energy load at several scales is a crucial task for grid managers and utilities, particularly with the rise of intermittent renewable energy sources demanding a heightened level of planning. In this project, we explore ways to predict energy demand at several levels of hierarchy in a grid, from individual meters to large substations serving entire neighborhoods. A desirable property in this context is that the sum of the load time series at the bottom of the hierarchy (e.g., individual meters) matches the load at higher levels. We propose two methods to achieve this hierarchical load forecasting objective: a two-step approach consisting of reconciling base learners and a deep learning-based end-to-end pipeline. We benchmark several design choices for these two methods and analyze whether they improve accuracy at various levels of the hierarchy. Finally, we compare them in terms of robustness and tractability on real massive datasets.

1 Introduction

Load forecasting, which consists of predicting the demand for energy consumption, is a critical mission in the Energy industry [1]. Indeed, grid managers have to constantly maintain a balance between supply and demand to avoid the risk of blackouts. Besides, supply is often highly inelastic, due to physical constraints linked to "shutting down" or "turning on" power plants, strengthening the importance of accurate demand forecasting. Historical load data also often integrates a hierarchical aspect. Utilities commonly have access to individual consumption data at the meter level and want to forecast aggregate energy consumption at various levels of groupings of individual meters (which could be neighborhoods, regions, etc.). Therefore, in this context, hierarchy refers to sum constraints between several time series representing different levels of aggregation of individual meters, which constitute the bottom level of the hierarchy.

Hierarchical forecasting is a relatively well-developed field in the Time Series literature. Two types of methods have been developed: a first range of methods consisting of reconciling predictions from base forecasts. From very basic approaches like Bottom-Up or Top-Down reconciliation [2], optimal methods have been developed such as Trace Minimization [3]. On the other hand, end-to-end deep learning approaches [4] have recently been developed to combine the time series forecasting model fitting with the hierarchical reconciliation step. In this report, we compare these two types of methods. For each one, we present several possible design choices and discuss their performance in terms of accuracy and scalability.

2 Dataset: GEFCom 2017 Competition

2.1 Dataset description

The GEFCom 2017 competition [5] by Hong et. al was a hierarchical load forecasting competition. The task was to provide probabilistic load forecasts at several levels of aggregation based on a large dataset. Teams' submissions were assessed using a holdout dataset encompassing one year of data following the period covered by the training dataset.

The dataset contains the following information:

- Seven years (from 2005 to 2011) of hourly load data measured at the level of 183 individual meters.
- Hierarchy information mapping the individual meters to intermediary stations aggregating the consumption of the meters. The hierarchical structure is presented in Figure 1.
- Hourly temperature and relative humidity data measured at 28 weather stations for which the distance with respect to the meters was not provided.

One of the greatest challenges for working with this dataset is its size. It gives access to a total of 10 million load readings and 3.4 million weather data points. This large scale generates constraints on the models that we can use, as we will see in Section 3.1.1.

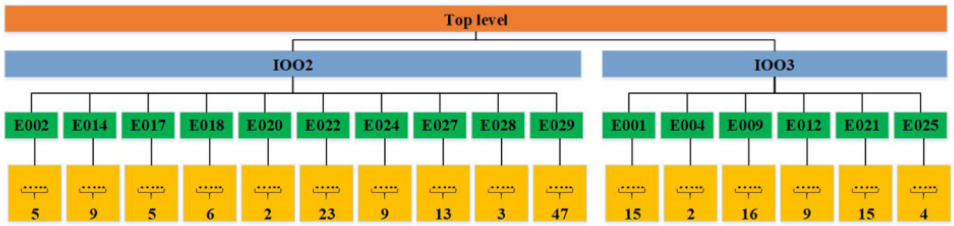


Figure 1: Hierarchical organization of the data, where the numbers at the bottom level indicate the number of meters (total of 183 meters at the base level).

2.2 Task, evaluation method and metrics

Based on the historical data from 2005 to 2010, the task is to forecast the load during the last year of data (2011), at all nodes in the hierarchy, from each of the individual meters themselves to the top level summing the readings of all the meters. In the context of load forecasting, where power plant production changes are constrained by inertia, we prioritize RMSE as the primary metric to significantly penalize large forecasting errors. We will also report Normalized RMSE as a convenient metric capturing the scale of the error compared to the scale of the test data. To quantify the performance of our models on each of the aggregation levels, we will report one RMSE and normalized RMSE for each level of the hierarchy.

3 Methodology

3.1 Method 1: Hierarchical reconciliation of base learners

As stated in the introduction, the Hierarchical Forecasting literature was first developed around a two-stage process: First, obtain base forecasts for some levels of the hierarchy. Then, reconcile them so that sum constraints between successive levels of the hierarchy are verified.

3.1.1 Base forecasts

To facilitate hierarchical reconciliation, we decided to train base learners for all time series of the hierarchy. We designed several base learners, that we benchmarked in Table 2 at the Sum and the Bottom level. We discuss here the main design choices that we made for each of these models:

Historical Averages We first implemented a very simple baseline that consists of estimating the load by averaging historical loads with the same *(month, day of the week, time of the day)* combination for each one of the considered grouping of meters.

SARIMA To choose the hyperparameters of our SARIMA base learners, we used the sum time series and first applied a first-order differentiation to remove the trend, which was sufficient to observe a zero mean in the process. The PACF and ACF plots indicated a 24-hour seasonality, as well as decays for seasonal and non-seasonal PACF and ACF values. Therefore, we fitted a SARIMA($p = 1, d = 1, q = 1$)x($P = 1, D = 1, Q = 1$) model with $s = 24$ on the scaled top time

series. We additionally used BIC to test other values of p, q, P, and Q. The values given above proved to be the ones that yielded the minimum BIC, namely achieving the best compromise between model complexity and performance.

SARIMAX A major problem in our context with the SARIMA model is that it only uses autoregressive patterns present in the time series and doesn't incorporate weather data or additional important covariates such as holidays, that have a high impact on load. We thus tried to train SARIMA models with exogenous covariates. However, the fitting process of these models is very computationally intensive, and fitting each base learner would take approximately one hour. Therefore, it was not realistic to train more than 200 base forecasts using these models. Besides, they would still capture the effects of covariates linearly, whereas there are clear non-linear effects in the data. For example, the load is high both for very high temperatures (high A/C usage) and low temperatures (high electric heating usage).

Prophet model with custom seasonality modeling Prophet is a time series forecasting library based on an additive model with non-linear yearly, weekly, daily, and holiday effects [6]. The seasonal components fitted by Prophet, shown in Figure 2, reveal the seasonal patterns present in the data. Load is higher during the week than on weekends, and July and August are the months of the year with the highest average load, which is probably due to intense A/C use. As the model proposed by Prophet by default didn't give satisfactory results, we added additional covariates to incorporate weather and holiday data, and we implemented cross-scale seasonality patterns. We had remarked the distribution of load during the day looks very different in the Winter, where two load peaks certainly correspond to the extra heating and appliance use when people are at home in the early morning and after 5 PM, and in the Summer, where AC is progressively turned on during the day as outdoor temperature rises. As a result, we modeled a time-of-the-day seasonality component that depends on the Season, which can be seen on the left part of Figure 2.

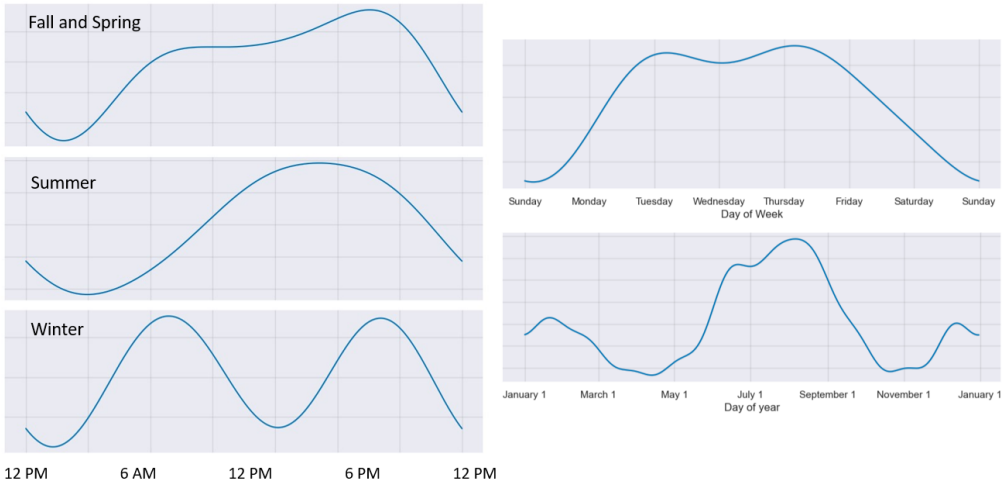


Figure 2: Seasonal components fitted by Prophet on the top time series. On the left, daily component depending on the Season that we modeled. On the right, weekly and monthly components.

LightGBM Finally, we trained *LightGBM* boosting-based models [7], as they are scalable enough to be quickly fitted on our large dataset with reasonable memory requirements, and allow us to model the complex non-linear dependency between load at each level, seasonality effects and weather. We concatenated the measured temperature and humidity values with the historical load data and used basic feature engineering to capture seasonality effects. We added cosine and sine transformations of the month and the hour, as we assume that close months and close times of the day will have the same load patterns, and we added an indicator variable of whether the day is a holiday in Massachusetts, the state the data was collected in. For each time series, we performed hyperparameter tuning using cross-validation and a search procedure with a fixed time budget. To limit the number of models to train at the bottom level of the hierarchy, we trained one model for each first-level grouping level. The underlying assumption of this choice is that meters that belong to the same mid-level grouping have

similar consumption patterns, which we had observed. We thus used one model for each mid-level aggregation level to predict the bottom time series, and each one of these models had the one-hot encoded version of the meter ID of the observed measurements as features.

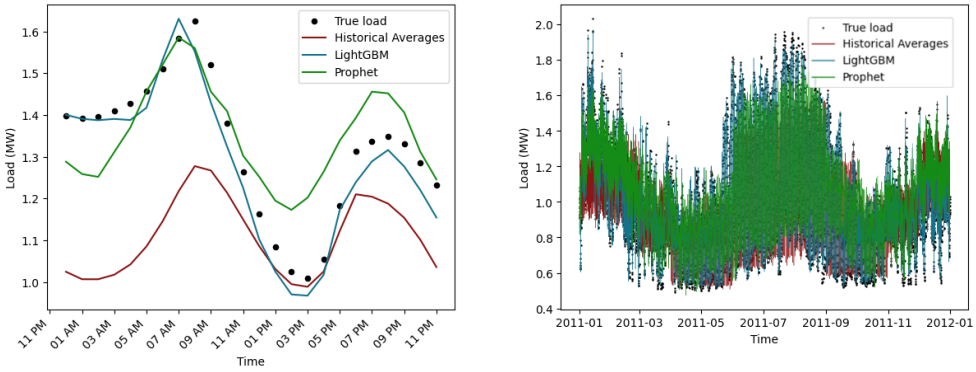


Figure 3: Comparison of load forecasts on the top time series on one day (1/15/2011, left) and on the whole test dataset (calendar year 2011, right).

3.1.2 Hierarchical reconciliation

Once trained, the predictions of these base learners, used for forecasting load across the hierarchy, will not be consistent with each other, resulting in the sum of lower-level loads not precisely matching higher-level loads in the hierarchy. To obtain more consistent, explainable, and hopefully more accurate forecasts, we need to apply a reconciliation method to enforce the sum constraints encoded by the hierarchy. We used three different methods to this extent.

Bottom-up The most simple one, the Bottom-up approach [2], consists of using only predictions at the bottom (meter) level and summing the results up to obtain forecasts for each grouping at higher hierarchy levels.

Top-down The top-down approach consists of fitting and predicting only the top time series (sum of all the meter-level time series) and distributing it among lower levels of hierarchy using the contributions $p_{j,l}$, with $y_{j,l} = p_j * y_{l+1}$ for the j-th sub-element of the grouping having time series (y_{l+1}). Several approaches are possible, but based on our experiments, the most effective is to use the proportion of historical averages:

$$p_{l,j} = \sum_{t=1}^{T} \frac{y_{j,l,t}}{T} / \sum_{t=1}^{T} \frac{y_{l+1,t}}{T} \tag{1}$$

Optimal reconciliation with MinTrace More generally, as outlined in [8], any reconciliation method can be written as finding an optimal mapping matrix G such that for all $h \in [|1, T|]$, the reconciled h-steps ahead forecast $\tilde{y}_h$ verifies $\tilde{y}_h = SG\hat{y}_h$, where $S \in \{0,1\}^{n \times m}$ is the aggregation matrix encoding the known hierarchical structure (where n is the number of nodes in the hierarchy and m is the number of time series at the bottom level), and $\hat{y}_h$ are the base forecasts for each node.

The objective is to minimize the variances of the coherent h-steps-ahead forecasts. With this structure, this variance can be written as $Tr(V_h)$, where the covariance matrix V_h is defined by $V_h = Var(y_{T+h} - \tilde{y}_h) = SGVar(y_{T+h} - \bar{y}_h)G^T S^T$. In addition, the reconciled forecasts should be unbiased, which imposes the constraint $SGS = S$. Overall, as proven in [3], given an approximation for $W_h = Var(y_{T+h} - \bar{y}_h)$, the covariance matrix of the base forecasts, this minimization problem, which is formalized below, has a unique solution.

$$\begin{aligned} \underset{G}{\text{minimize}} \quad & \text{Tr}(SGW_hG^TS^T) \\ \text{s.t.} \quad & SGS = S \end{aligned}$$

Several choices are possible to approximate the covariance matrix of the base forecasts W_h:

- *OLS approximation*: The most simplifying solution is to set $W_h = k_h I$ for all h, with k_h a positive constant for each time step [3]. In this context, G is made independent of the data and the solution to the optimization problem is the least squares estimator of $\hat{y}$ against S.
- *WLS with variance scaling* (WLS-v): This approximation consists of setting $W_h = k_h \text{diag}(\hat{W}_1)$, where $k_h > 0$ and $\hat{W}_1 = \frac{1}{T}\sum_{t=1}^{T} e_t e_t^T$, where e_t are the residuals of the one-step-ahead predictions on the train set. The name of this approach comes from the fact that it scales the base forecasts using the variance of their residuals.
- *WLS with structural scaling* (WLS-s): We set $W_h = k_h \Lambda$, where $k_h > 0$ and $\Lambda = \text{diag}(S\mathbf{1})$. This formulation assumes that forecast errors at the lowest level are uncorrelated and all have variance k_h. The diagonal elements of the matrix Λ represent the number of forecast error variances contributing to each node higher in the hierarchy. This estimator relies on the structure of aggregations instead of on the actual data, which explains its denomination of *structural scaling*.

All these possible choices are benchmarked on our dataset in Table 4, and interpretations on which methods are the most adapted to our setting are provided in 4.2.

3.2 Method 2: Integrated End-to-End Deep Learning Approach

The integrated End-to-End Approach [4] produces coherent probabilistic forecasts without requiring any explicit post-processing step. The method simultaneously learns from all of the time series in the hierarchy and incorporates the reconciliation step as part of a single trainable model. This is achieved by using the observation that reconciliation can be cast as an optimization problem with a closed-form solution. This model specification makes simultaneous learning of hierarchical and reconciled forecasts possible while accomplishing the task of generating forecasts that are both probabilistic and coherent.

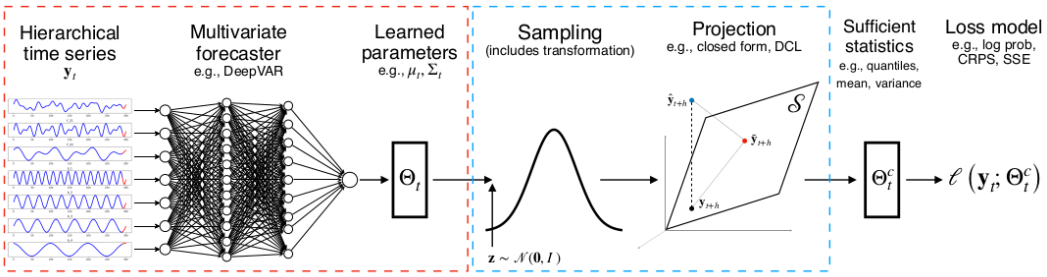


Figure 4: Hierarchical DeepVar Model Architecture. Hierarchical time series data is used to train a multivariate forecaster (DeepVar). Learned distribution parameters allow this distribution to be sampled during training. Samples are then projected to enforce coherency. The loss is computed after sampling from the distribution.

In practice, the multivariate forecaster consists of a recurrent neural network with 2 layers of LSTM cells. The number of input features of the LSTM is equal to the number of bottom time series in the hierarchical data. To achieve reconciliation, we use the following mathematical approach:
$y_t = Sb_t$, with S the hierarchy matrix, b_t the bottom time series, and y_t the grouped time series above this bottom level in the hierarchy. $y_t = Sb_t \Leftrightarrow \begin{bmatrix} a_t \\ b_t \end{bmatrix} = \begin{bmatrix} S_{sum} \\ I_m \end{bmatrix} b_t$. This problem can be re-written as $Ay_t = 0$ with $A = [I_r | -S_{sum}]$. We are thus solving the problem $\hat{y}_t = \underset{y \in \mathbb{R}^n}{\arg\min} \|\bar{y} - y\|_2 \; s.t. Ay = 0$.
We can solve this problem in a closed form solution by projecting the base learner's forecasts with $M = I - A^T(AA^T)^{-1}A$. The forecasting steps consist in obtaining forecast distribution parameters by unrolling the RNN for one time step using the last hidden state from training h_t, generating a set of sample predictions by taking Monte Carlo samples from the distribution parameters and projecting them with the same matrix M used during training. Table 1 summarizes the choices we made to implement this model.

Model	Parameter
Cell Type	LSTM
Number of cells	20
Number of layers	2
Number of samples	100
Dropout rate	0.1

Table 1: DeepVar Architecture choices

4 Results

4.1 Base forecasts benchmark

We evaluate the performance of our base forecasts in Table 2, which outlines the accuracy of each base learner to predict the top time series (sum of all meter loads) and the first bottom time series (meter with ID 1) with a time horizon of one year ($H = 8760$). The predictions of these models on the Sum time series are visually represented in Figure 3.

	Sum		Bottom - First meter	
Model	**RMSE**	**nRMSE**	**RMSE**	**nRMSE**
LightGBM	**45,998**	**4.5%**	**440**	**12.8%**
Prophet	161,943	15.9%	946	12.8%
Historical Averages	175,257	17.2%	954	27.8%
SARIMA	469,379	46.0%	1,629	47.4%

Table 2: Base forecasts accuracy on the sum time series and the first meter (bottom time series) for the whole year 2011

As can be seen, the best-performing model on both the Sum and the first Bottom time series is our *LightGBM* model. The large performance delta with the other models can be explained by the fact that SARIMA only models autoregressive patterns in the time series, the Prophet model models the impact of the exogenous weather covariates linearly, and the historical averages baseline is likely largely under-fitting. On the other hand, the *LightGBM* base learners successively take into account the highly non-linear relations with the weather data or with the seasonality effects described in Figure 2. The hyper-parameter tuning that we realized based on cross-validation and the nature of the model based on boosting allowed to capture these complex effects while limiting overfitting.

4.2 Impact of hierarchical reconciliation with the two-step approach

Table 4 outlines the impact of the different reconciliation methods presented in 3.1.2 to reconcile the base forecasts obtained by our *LightGBM* models. The RMSE and normalized RMSE are aggregated on each level of the hierarchy. The "No reconciliation" approach consists of fitting and predicting each one of the bottom, middle, top, and sum time series, without applying any reconciliation method.

Reconciliation method	**Sum**		**Top**		**Middle**		**Bottom**	
	RMSE	**nRMSE**	**RMSE**	**nRMSE**	**RMSE**	**nRMSE**	**RMSE**	**nRMSE**
No reconciliation	45,998	4.5%	30,728	6.1%	5,753	8.6%	1,679	26.6%
Bottom-up	45,078	4.4%	29,226	5.8%	6,069	9.1%	1,679	26.6%
Top-down	45,998	4.5%	29,980	6.0%	5,906	8.8%	1,690	26.8%
MinTrace - OLS	45,363	4.4%	29,987	6.0%	5,824	4.5%	1,675	26.6%
MinTrace - WLS-v	**44,248**	**4.3%**	**28,725**	**5.7%**	5,757	8.6%	**1,661**	**26.3%**
MinTrace - WLS-s	44,410	4.4%	28,865	5.8%	**5,726**	**8.5%**	1,671	26.5%

Table 3: Comparison of hierarchical reconciliation methods with *LightGBM* base forecasts on the whole test set. When looking at Figure 1, the Sum column corresponds to the orange top-level, Top column to the "I00" blue level, Middle to "E0" green level, and Bottom to the yellow (meter) level.

The key finding from this table is that reconciling forecasts is always beneficial for accuracy, regardless of the level in the hierarchy. The RMSE given by the reconciled forecasts is indeed well below the one given by the base forecasts at the four hierarchy levels visible in Figure 1. The interpretation of this performance gain is that reconciling the forecasts allows to regularize the base forecasts and is very useful in limiting overfitting. The constraint that the bottom time series must sum to the top time series is a supplementary information that we have on the test set, that we can use to improve test accuracy. This interpretation matches what can be seen on Figure 5a: the reconciled curves: both at the upper and lower levels of the hierarchy, look slightly more "smooth" on the test set, indicating that the reconciliation is a useful regularization step to limit overfitting.

Among hierarchical reconciliation methods, as expected, MinTrace-based methods yield better accuracy. The reconciled forecasts are thus guaranteed to be unbiased, conversely to the ones obtained using the Bottom-Up and Top-Down methods, and optimally reduce the variance of the h-steps ahead reconciled forecasts for any $h \in [|1, H|]$. Among MinTrace-based methods, WLS, and in particular WLS-v, performs best. This was expected as the WLS-v approximation to the covariance matrix W_h is the one that incorporates the most information from the training data. It scales the base forecasts using the variance of the residuals of the test set, therefore taking into account which forecasts we are the most uncertain about in the reconciliation process. Conversely, the OLS approximation didn't take into consideration the differences in scale between the different levels of the hierarchy and WLS-s was only using the aggregation structure without exploiting the variance of the forecasts on the train set.

The last observation that we can make is that accuracy becomes poorer in comparison to the scale of the data (as measured by the normalized RMSE) when going lower in the hierarchy. Several interpretations can account for this phenomenon. First, the *LightGBM* base models are pooled among the first hierarchy level for predicting the load at each individual meter for computational efficiency, as explained in 3.1.1. Besides, the load at the bottom level is the most unpredictable. Whereas aggregate levels are smoothed and largely dependent on seasonality and weather-related consumption patterns, consumption at the bottom level is highly dependent on the particular activities of each household. Yet we don't have any data about the location of individual meters or information about these households. In these conditions, WLS-v is a very useful method to incorporate this higher uncertainty.

4.3 Impact of hierarchical reconciliation with the end-to-end approach

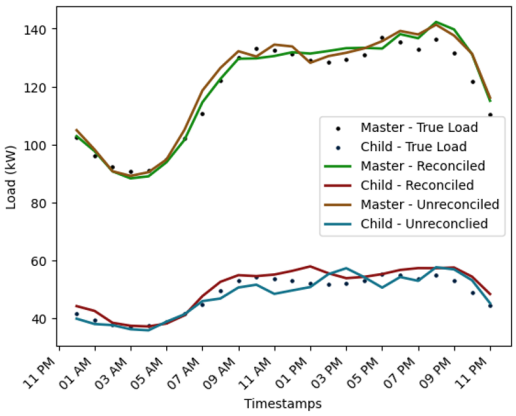
Level of hierarchy	Sum	Top	Middle	Bottom
DeepVar no rec.	97,913	49,010	5,403	6,923
Hierarchical DeepVar	**97,901**	**48,987**	**5,402**	**6,791**

Table 4: RMSE of DeepVar without reconciliation and Hierarchical DeepVar models at each level of the hierarchy. The forecasting horizon is equal to 85 (3 days)

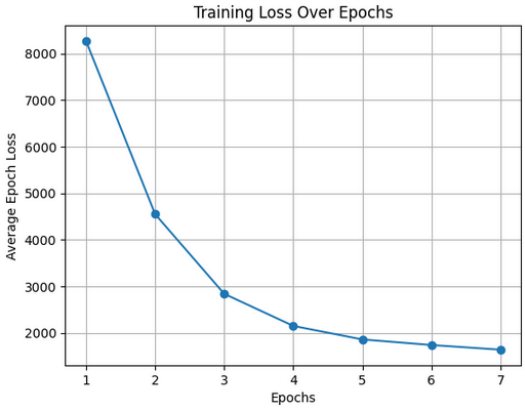
Our benchmarking capacity is significantly constrained due to the high computational demands associated with concurrently fitting 183 base series, training our model on six years of load data, and making predictions at an hourly frequency. Notably, the forecasting horizon poses a specific limitation, as we were unable to extend predictions beyond a 3-day span while operating within the constraints of a 20GB RAM capacity. Despite these computational challenges preventing us from benchmarking DeepVar against other forecasting models considered in this study, we conducted a comparative analysis between the Hierarchical DeepVar model and a conventional DeepVar LSTM model trained without hierarchical information. In scenarios with similar training and testing conditions, the hierarchical version of DeepVar consistently outperformed the simpler DeepVar approach across all hierarchy levels. This observation suggests that, even in a constrained testing environment, the deep learning one-step hierarchical approach yields superior performance compared to a non-reconciliation approach.

5 Conclusion and future work

To conclude, two approaches are possible to generate coherent load forecasts on a large dataset such as the GEFCom 2017 competition dataset: a two-step approach consisting of fitting, then



(a) Unreconciled and *WLS-s* reconciled LightGBM predictions on 4/9/2011 for a mid-level station and one of its child individual meters



(b) Hierarchical DeepVar Training curve loss

Figure 5: Impact of hierarchical reconciliation

reconciling, base forecasts, and an integrated end-to-end deep learning approach relying on sampling and projection.

For both methods, important design choices have to be made in order to obtain the best possible predictions. Given the scale and the nature of the data, it is preferable to use a scalable boosting model like *LightGBM* with appropriate engineered features as the base forecasts. Hierarchical reconciliation should be performed to regularize the best forecasts using the MinTrace method with the WLS approximation with variance scaling. For the integrated deep learning approach, careful architecture choices should be made on the cell type, layer dimensions, and dropout rates.

We can see several directions for further research on Hierarchical load forecasting on a large dataset:

First, our autoregressive base learners didn't match the performance of our boosting model with engineered features, due to the length of the test set and the importance of the exogenous covariates. As base learners, we could use deep recurrent networks or transformers that take both previous series values and exogenous features into consideration.

If base learners are already sufficiently regularized, it might not be relevant to reconcile them to improve test accuracy at each level of the hierarchy. In the future, we could compare the impact of the hierarchical reconciliation step on several sets of base forecasts.

The end-to-end deep learning approach offers an effective solution by consolidating the two-stage procedures into a single step, resulting in better performance across all levels compared to scenarios without reconciliation. However, implementing this method on real-world datasets proves to be challenging. A logical progression involves exploring alternative architectures beyond DeepVar to achieve long-term forecasting results. Initial attempts with raw LSTMs failed to deliver successful forecasts over a year horizon at an hourly frequency, suggesting that for extended forecasting periods, more specific Deep Learning models that decompose time series into trend and seasonality components may be more pertinent.

Finally, in the deep learning setting, our findings indicate that overall performance is enhanced by enforcing coherency. A valuable next step would involve a more detailed examination of the impact of reconciliation by introducing a loss on the base learners with a penalty parameter for coherency. By systematically varying the coherency penalization parameter, we could gain nuanced insights into the precise impact of reconciliation on model performance.

References

[1] Hesham K Alfares and Mohammad Nazeeruddin. Electric load forecasting: literature survey and classification of methods. *International journal of systems science*, 33(1):23–34, 2002.

[2] Chul-Yong Lee and Sung-Yoon Huh. Forecasting new and renewable energy supply through a bottom-up approach: The case of south korea. *Renewable and Sustainable Energy Reviews*, 69:207–217, 2017.

[3] George Athanasopoulos Shanika L. Wickramasuriya and Rob J. Hyndman. Optimal forecast reconciliation for hierarchical and grouped time series through trace minimization. *Journal of the American Statistical Association*, 114(526):804–819, 2019.

[4] Syama Sundar Rangapuram, Lucien D Werner, Konstantinos Benidis, Pedro Mercado, Jan Gasthaus, and Tim Januschowski. End-to-end learning of coherent probabilistic forecasts for hierarchical time series. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8832–8843. PMLR, 18–24 Jul 2021.

[5] Tao Hong, Jingrui Xie, and Jonathan Black. Global energy forecasting competition 2017: Hierarchical probabilistic load forecasting. *International Journal of Forecasting*, 35(4):1389–1399, 2019.

[6] Sean J Taylor and Benjamin Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

[7] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30, 2017.

[8] Rob J Hyndman, Roman A Ahmed, George Athanasopoulos, and Han Lin Shang. Optimal combination forecasts for hierarchical time series. *Computational statistics & data analysis*, 55(9):2579–2589, 2011.