Hierarchical Time Series Forecasting Via Latent Mean Encoding

Alessandro Salatiello
Zalando SE
alessandro.salatiello@uni-tuebingen.de

Stefan Birr
Zalando SE
stefan.birr@zalando.de

Manuel Kunz
Zalando SE
manuel.kunz@zalando.de

Abstract

Coherently forecasting the behaviour of a target variable across both coarse and fine temporal scales is crucial for profit-optimized decision-making in several business applications, and remains an open research problem in temporal hierarchical forecasting. Here, we propose a new hierarchical architecture that tackles this problem by leveraging modules that specialize in forecasting the different temporal aggregation levels of interest. The architecture, which learns to encode the average behaviour of the target variable within its hidden layers, makes accurate and coherent forecasts across the target temporal hierarchies. We validate our architecture on the challenging, real-world M5 dataset and show that it outperforms established methods, such as the TSMixer model.

1 Introduction

Hierarchical forecasting studies methods to ensure coherent forecasts across the inherent hierarchies in data [Hyndman et al., 2011]. It is a growing research field, particularly relevant in domains such as e-commerce [Rangapuram et al., 2023, Sprangers et al., 2024] and electricity demand forecasting [Taieb et al., 2017]. In these domains, cross-sectional hierarchies, such as geographical hierarchies or product category trees in retail demand forecasting [Kunz et al., 2023], are naturally present in the data. Similarly, temporal hierarchies arise from the need to provide coherent forecasts on multiple time resolutions [Taieb, 2017]. In retail demand forecasting, for example, high-resolution forecasts are required to steer inventory on a daily or weekly frequency. In contrast, strategic planning and finance need forecasts on a quarterly level for geographical regions or product categories [Sprangers et al., 2024]. Although structural and temporal hierarchies can be intertwined [Kourentzes and Athanasopoulos, 2019], here we focus on providing coherent forecasts along the temporal dimension. This problem is often addressed by training independent, specialized forecasting models for each frequency of interest, followed by a post-processing reconciliation step to ensure coherence [Kourentzes et al., 2014]. Alternatively, established methods [Athanasopoulos et al., 2017] avoid the post-processing step by using a bottom-up approach that only requires training a single model at the observed, fine-grained frequency. With this work, we provide a novel approach to coherent temporal hierarchical forecasting based on using a single hierarchical neural network model with frequency-specific modules. Importantly, the network — trained end-to-end on a state-of-the-art deep learning architecture — does not require post-processing steps and outperforms established forecasting models on real-world data.

NeurIPS 2024 Workshop on Time Series in the Age of Large Models.

2 Methods

In this work, we set out to develop an efficient architecture for coherent time series forecasting at multiple temporal aggregation levels. More formally, we aim to address the problem of simultaneously and coherently forecasting the future behaviour of a target variable $y(t)$ at its original base sampling frequency and at a coarser frequency over a forecasting horizon of h samples. Specifically, we are interested in the coarse-grained time series $y(t)^{avg}$ obtained from the fine-grained one $y(t)$ by computing non-overlapping averages over k bins of w samples. Thus, in summary, our architecture should be able to forecast $y_{t+1:t+h} := \{y_{t+i}\}_{i=1}^{h}$ and $y_{t,1:k}^{avg} := \{y_{t,j}^{avg}\}_{j=1}^{k}$ with $k = \frac{h}{w}$, respecting the following constraint:

$$y_{t,j}^{avg} = \frac{1}{w} \sum_{i=1}^{w} y_{t+i+w(j-1)} \tag{1}$$

The predictions can be based on the past and future values of both categorical covariates $x_{t-c+1:t+h}^{cat}$ and continuous covariates $x_{t-c+1:t+h}^{cont}$, over a window extending c samples into the past and h samples into the future. Importantly, the covariates can include the past true values of the target variable $y_{t-c+1:t}$ as well as static variables.

2.1 The encoder-decoder architecture

The architecture we developed to tackle this problem is represented in Fig.1. The architecture consists of three trainable modules — the embedder $b(z; \psi)$, the encoder $e(z; \theta)$, and the decoder $d(z; \phi)$ modules — which work synergistically to solve the hierarchical forecasting problem. Indeed, our design encourages the encoder module to specialize in forecasting the coarse-grained time series while the decoder module specializes in forecasting the fine-grained time series. Meanwhile, the

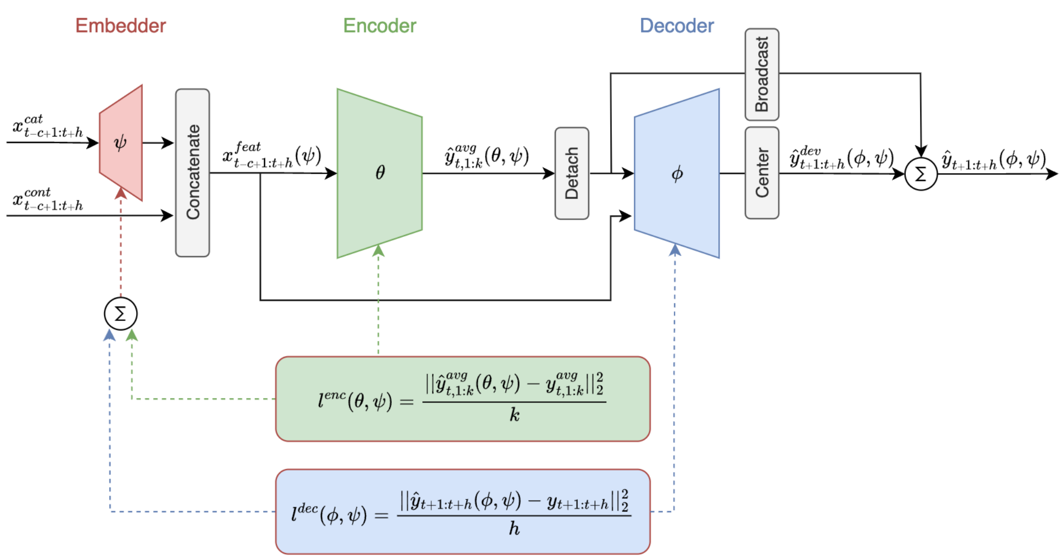

Figure 1: The proposed architecture. The architecture forecasts h future values of a target variable y using c past and h future values of categorical x^{cat} and continuous x^{cont} covariates. The architecture comprises three main modules — the embedder, the encoder, and the decoder modules — as well as a non-trainable readout layer. The embedder module learns a continuous representation of the categorical variables. The encoder module learns to predict $\hat{y}_{t,1:k}^{avg}(\theta, \psi)$: the average values of the target variable over k consecutive bins of size w. The decoder learns to predict $\hat{y}_{t+1:t+h}^{dev}(\phi, \psi)$: the deviations from the average values. Finally, a non-trainable readout layer computes the architecture output $\hat{y}_{t+1:t+h}(\phi, \psi)$ by summing aligned representations of the averages and deviations. The architecture is trained by minimizing two loss functions: the encoder loss $l^{enc}(\theta, \psi)$, which optimizes the encoder and the embedder, and the decoder loss $l^{dec}(\phi, \psi)$, which optimizes the decoder and the embedder.

embedder module specializes in learning continuous representations of the categorical covariates that are useful for forecasting both the coarse and fine-grained behaviour of the target variable.

More precisely, the embedder module embeds the categorical variables $x^{cat}_{t-c+1:t+h}$ into a dense, multidimensional continuous space; these are then concatenated with the continuous variables into a tensor $x^{feat}_{t-c+1:t+h}(\psi)$ and forwarded to both the encoder and decoder modules. The encoder module learns to predict the average values of the target variable over k consecutive bins of size w: $\hat{y}^{avg}_{t,1:k}(\theta, \psi)$. These average values are then forwarded to the decoder, which can, in turn, learn to predict the deviations from the average values $\hat{y}^{dev}_{t+1:t+h}(\phi, \psi)$. This is enforced by postprocessing the decoder output with a centering module that subtracts the average values the decoder outputs over the k bins of size w.

Finally, a non-trainable readout layer computes the architecture output $\hat{y}_{t+1:t+h}(\phi, \psi)$ by summing aligned representations of the averages and the deviations according to:

$$\hat{y}_{t+1:t+h}(\phi, \psi) = \mathbf{S}_{w,k}\hat{y}^{avg}_{t,1:k}(\theta, \psi) + \hat{y}^{dev}_{t+1:t+h}(\phi, \psi), \tag{2}$$

with $\mathbf{S}_{w,k} = (s_{i,j})_{1\leq i\leq h,\, 1\leq j\leq k}$ defined by:

$$s_{i,j} = \begin{cases} 1 & \text{if } (j-1)w < i \leq jw, \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The architecture is trained by minimizing two loss functions. The encoder loss, which is used to optimize the encoder and the embedder modules:

$$l^{enc}(\theta, \psi) = \frac{||\hat{y}^{avg}_{t,1:k}(\theta, \psi) - y^{avg}_{t,1:k}||^2_2}{k} \tag{4}$$

and the decoder loss, which is used to optimize the decoder and the embedder modules:

$$l^{dec}(\phi, \psi) = \frac{||\hat{y}_{t+1:t+h}(\phi, \psi) - y_{t+1:t+h}||^2_2}{h} \tag{5}$$

As backbone networks for the encoder and decoder modules, we used untrained Time-Series Mixer (TSMixer) models Chen et al. [2023][1], due to their proved efficiency at extracting temporal and cross-variate patterns stacking simple time-mixing and feature-mixing MLP layers. However, the choice of the backbone network is flexible and not constrained by the architecture.

Further training details are provided in the appendix, in section A.0.2.

2.2 The dataset

To validate the proposed architecture, we chose the challenging M5 dataset: the real-world, large-scale dataset used in the M5 forecasting competition Makridakis et al. [2022]. The dataset comprises 30,490 time series representing the number of retail sales of products sold in ten US stores over a period of six years. The features we used, the cardinality of the categorical variables, and their corresponding number of embedding dimensions are provided in the appendix in Table 2 and Table 3.

2.3 Evaluation

The M5 dataset contains sales data over a period of 1942 days. Following common practice (e.g., see Chen et al. [2023]), we used the first 1886 days for training, the next 28 days for validation, and the last 28 days for testing. Similarly to Chen et al. [2023], we chose a context window $c = 35$ and a forecasting horizon $h = 28$. As the forecasting horizon includes four weeks, we chose a binning size of a week (i.e., $w = 7$ and thus $k = 4$). Our main performance measure is the one used by the M5 competition to rank the submissions: the weighted root mean squared scaled error (WRMSSE[2] — see the appendix for a formal definition). However, we also consider more traditional metrics such as the root mean squared error measured at the fine-grained, daily frequency (RMSEd), and at the coarse-grainded, weekly frequency (RMSEw). Additionally, we considered the RMSE of the residuals (RMSEr), that is, the one of the daily-level time series after subtracting the estimated average over the binning windows. Finally, we also considered the median fraction of explained variance (MFEV) and the mean absolute deviation (MAD).

[1]https://github.com/ditschuk/pytorch-tsmixer
[2]https://github.com/pmrgn/m5-wrmsse

Table 1: Results. The table contains the performance of the considered models on the test split. For each performance metric, we indicate the value of the best-performing model with a **bold** font style and the value of the second-best model with an underlined font style. Note that the metric values of the models with a "‡" superscript are taken from Chen et al. [2023]. Also note that our models were trained for a maximum of 100 epochs, whereas the models in Chen et al. [2023] were trained for up to 300 epochs. NBNLL indicates the negative binomial negative log-likelihood loss, while MSE indicates the mean squared error loss. We refer to section 2.3 for details on the considered metrics and to section 2.4 for details on the benchmark models.

Model	Loss	Metric					
		WRMSSE (↓)	RMSEd (↓)	RMSEw (↓)	RMSEr (↓)	MFEV (↑)	MAD (↓)
CtxWindAVG		1.085	2.242	1.316	1.815	0.000	1.066
DeepAR‡	NBNLL	0.789					
TFT‡	NBNLL	0.670					
TSMixer-Ext‡	NBNLL	0.640					
MonoNB	NBNLL	0.681	2.179	1.269	1.771	0.024	1.053
MonoMSE	MSE	0.672	2.192	1.330	1.742	0.026	1.058
EncDecNB	NBNLL	0.634	**2.091**	**1.178**	**1.728**	0.027	**1.035**
EncDecMSE	MSE	**0.620**	2.146	1.263	1.735	**0.028**	1.051

2.4 Benchmark models

Our main comparison is against the monolithic TSMixer architecture (Mono), which was shown to outperform several established forecasting models on popular datasets, including the M5 dataset [Chen et al., 2023]. Critically, for a fair comparison, we kept the number of parameters comparable by splitting the number of hidden units of the monolithic architecture ($n_h = 64$) between the encoder and the decoder modules (i.e., $n_e^{enc} = 32$ and $n_e^{dec} = 32$). We also compare our architecture (EncDec) against established time series forecasting models such as the Temporal Fusion Transformer (TFT — Lim et al. [2021]) and the Deep Autoregressive Recurrent Network (DeepAR —Salinas et al. [2020]), and against a simple naive model which always forecasts the average over the context window (CtxWindAVG).

3 Results

The results — summarized in Table 1 — indicate that the encoder-decoder architectures (EncDecNB and EncDecMSE) outperform the monolithic ones (MonoNB and MonoMSE) across all the considered metrics. The results also show that the encoder-decoder architectures outperform established forecasting models (DeepAR, TFT, and TSMixer-Ext) in terms of WRMSSE despite being trained for a lower number of epochs (100 vs 300 epochs).

Interestingly, in contrast to previous works [Chen et al., 2023, Salinas et al., 2020] that reported large performance boost when using the negative binomial negative log-likelihood (NBNLL) loss to train models on demand data characterized by bursty behaviour and widely varying magnitudes (similar to the M5 dataset), we were able to train models equally effectively with both the NBNLL and the MSE losses. As a matter of fact, in our case, we observed that MSE-trained models tend to outperform NBNLL-trained ones on two of the six considered metrics, namely the WRMSSE and MFEV metrics. This seems to suggest that MSE-trained models are better at capturing the behaviour of the time series with larger magnitudes and variances, while, conversely, NBNLL-trained models are better at capturing the behavior of the time series with smaller magnitudes and variances.

Finally, we note that the encoder-decoder architecture presented in this work was primarily designed to address the challenge of temporal hierarchical forecasting. As such, improved performance at the coarse, weekly level (RMSEw) was an expected outcome of a competent method. However, the general performance boost shown by our architecture suggests that the inductive bias we enforced is conducive to overall improved forecasting accuracy. Particularly noteworthy is, perhaps, the improvement in WRMSSE, which points to better performance in cross-sectional hierarchical forecasting.

References

George Athanasopoulos, Rob J Hyndman, Nikolaos Kourentzes, and Fotios Petropoulos. Forecasting with temporal hierarchies. *European Journal of Operational Research*, 262(1):60–74, 2017.

Si-An Chen, Chun-Liang Li, Nate Yoder, Sercan O Arik, and Tomas Pfister. Tsmixer: An all-mlp architecture for time series forecasting. *arXiv preprint arXiv:2303.06053*, 2023.

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.

Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Nikolaos Kourentzes and George Athanasopoulos. Cross-temporal coherent forecasts for australian tourism. *Annals of Tourism Research*, 75:393–409, 2019.

Nikolaos Kourentzes, Fotios Petropoulos, and Juan R Trapero. Improving forecasting by estimating time series structural components across multiple frequencies. *International Journal of Forecasting*, 30(2):291–302, 2014.

Manuel Kunz, Stefan Birr, Mones Raslan, Lei Ma, and Tim Januschowski. Deep learning based forecasting: a case study from the online fashion industry. In *Forecasting with Artificial Intelligence: Theory and Applications*, pages 279–311. Springer, 2023.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4): 1748–1764, 2021.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4):1346–1364, 2022. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.11.013. URL `https://www.sciencedirect.com/science/article/pii/S0169207021001874`. Special Issue: M5 competition.

Syama Sundar Rangapuram, Shubham Kapoor, Rajbir Singh Nirwan, Pedro Mercado, Tim Januschowski, Yuyang Wang, and Michael Bohlke-Schneider. Coherent probabilistic forecasting of temporal hierarchies. In *International Conference on Artificial Intelligence and Statistics*, pages 9362–9376. PMLR, 2023.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International journal of forecasting*, 36(3): 1181–1191, 2020.

Olivier Sprangers, Wander Wadman, Sebastian Schelter, and Maarten de Rijke. Hierarchical forecasting at scale. *International Journal of Forecasting*, 2024.

Souhaib Ben Taieb. Sparse and smooth adjustments for coherent forecasts in temporal aggregation of time series. In *NIPS 2016 Time Series Workshop*, pages 16–26. PMLR, 2017.

Souhaib Ben Taieb, James W Taylor, and Rob J Hyndman. Coherent probabilistic forecasts for hierarchical time series. In *International Conference on Machine Learning*, pages 3348–3357. PMLR, 2017.

Table 2: Features table. All the features we provide to the network are listed here. Note that the network receives in input only the c past values of the daily_sales; for all the other dynamic continuous variables, $c + h$ values are instead provided.

Features			
Static		Dynamic	
Categorical	Continuous	Categorical	Continuous
item_id store_id department_id category_id state_id	avg_price avg_nsales	weekday month cultural_event sporting_event religious_event national_event snap_day	daily_sales (c) daily_price year

Table 3: Feature cardinality table. Note that the number of embedding dimensions was determined according to the formula: $n_{emb} = \lceil 6r^{0.25} \rceil$. The binary variables were not embedded as we did not find this to be beneficial.

Categorical Features					
Static			Dynamic		
Name	Cardinality	Num. Emb. Dim.	Name	Cardinality	Num. Emb. Dim.
item_id	3049	45	weekday	7	10
store_id	10	11	month	12	12
department_id	7	10	cultural_event	2	1
category_id	3	8	sporting_event	2	1
state_id	3	8	religious_event	2	1
			national_event	2	1
			snap_day	2	1

A Appendix

A.0.1 Input features details

In this section, we provide the input features we used in Table 2. We also provide the cardinality of the categorical variables and their corresponding number of embedding dimensions in Table 3.

A.0.2 Training details

The encoder-decoder architecture was trained using two loss functions: the encoder loss $l^{enc}(\theta, \psi)$ — which is used to compute the gradients of the encoder $e(z; \theta)$ and the embedder $b(z; \psi)$ — and the decoder loss $l^{dec}(\phi, \psi)$ — which is used to compute the gradients of the decoder $d(z; \phi)$ and the embedder $b(z; \psi)$. More specifically, while the encoder and decoder modules are optimized using their respective loss functions, the embedder is optimized using the sum of these loss functions; that is:

$$\nabla_{\theta}^{enc} := \nabla_{\theta} l^{enc}(\theta, \psi) \tag{6}$$

$$\nabla_{\phi}^{dec} := \nabla_{\phi} l^{dec}(\phi, \psi) \tag{7}$$

$$\nabla_{\psi}^{emb} := \nabla_{\psi} l^{enc}(\theta, \psi) + \nabla_{\psi} l^{dec}(\phi, \psi) \tag{8}$$

For each training step, we perform a single forward pass through the model to compute the coarse-grained predictions $\hat{y}_{t,1:k}^{avg}(\theta, \psi)$ and the fine-grained ones $\hat{y}_{t+1:t+h}(\phi, \psi)$. Importantly, the decoder computes the fine-grained predictions based on a detached copy of the coarse-grained predictions (represented by the "detach" block in Fig.1). This ensures that the fine-grained predictions, and thus the decoder loss and the decoder gradients, do not depend on the encoder parameters θ.

Table 4: Hyperparameters table. The table reports the hyperparameters of the models shown in Table 1. All models were trained for a maximum of 100 epochs, without dropout, and with gradient clipping with a threshold value set to 1.

Model	Hyperparameters					
	Name Search Space	Loss	Loss Rescaling {Yes, No}	Learning Rate {1e-4, 1e-3, 4e-3}	Hidden Units	TSM Blocks
MonoNB		NBNLL	Yes	1e-3	64	2
MonoMSE		MSE	No	1e-3	64	2
EncDecMB		NBNLL	No	4e-3	32+32	2+2
EncDecMSE		MSE	No	1e-4	32+32	2+2

The training hyperparameters are provided in Table 4. We reused most of the hyperparameters specified in Chen et al. [2023] and performed a grid search for the learning rate and the loss-rescaling option, which specifies whether to rescale the time series to their original range before computing the loss. All models were trained using the Adam [Kingma, 2014] optimizer. The minibatches had size 30 and contained random items sampled by a weighted sampler with weights proportional to the average number of sales [Salinas et al., 2020]. The best model configurations were selected based on RMSEd computed on the validation set.

A.0.3 The weighted root mean squared scaled error (WRMSSE)

The main metric we used to evaluate the performance of our models is the WRMSSE: the metric used to rank the performance of the point forecasts in the M5 Forecasting Accuracy Competition[3]. WRMSSE aims to measure the ability of a model to coherently forecast data structured in cross-sectional hierachies such as those in the M5 dataset, which contains 30,490 time series of unit sales that can be grouped in 12 aggregation levels, for a total of 42,840 time series.

Specifically, WRMSSE is defined by:

$$WRMSSE = \sum_{i=1}^{42{,}840} w_i \times RMSEE_i \tag{9}$$

with:

$$RMSEE = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}\left(y_t - \hat{y}_t\right)^2}{\frac{1}{n-1}\sum_{t=2}^{n}\left(y_t - y_{t-1}\right)^2}} \tag{10}$$

In the equation above, h is the forecasting horizon, and n is the number of training days. Thus, RMSSE scales the RMSE of the test predictions with respect to the RMSE of the training predictions of a naive random walk model. Therefore, WRMSSE is a weighted average of RMSSE computed for all the atomic and aggregated time series in the dataset, accurately tracking the ability of a model to make accurate forecasts at each level of the hierarchy. Importantly, the weights w_i are chosen to be proportional to the total sales volume of the atomic and aggregated time series, which is estimated from the sales recorded in the validation window.

[3] https://github.com/Mcompetitions/M5-methods/blob/master/M5-Competitors-Guide.pdf