Forking-Sequences

Willa Potosnak, Malcolm Wolff, Boris Oreshkin, Mengfei Cao,
Amazon, New York, USA
{wpotosna, wolfmalc, oreshkin, mfcao}@amazon.com

Michael W. Mahoney, Dmitry Efimov, Kin G. Olivares
Amazon, New York, USA
{zmahmich, defimov, kigutie}@amazon.com

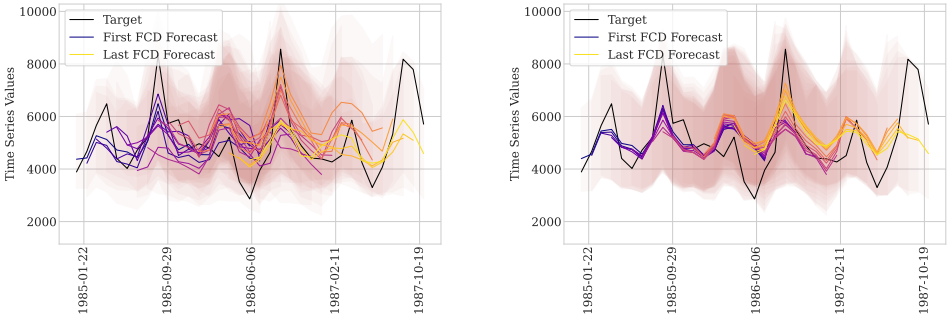
Abstract

While accuracy is a critical requirement for time series forecasting models, an equally important (yet often overlooked) desideratum is forecast stability across forecast creation dates (FCDs). Even highly accurate models can produce erratic revisions between FCDs, undermining stakeholder trust and disrupting downstream decision-making. To improve forecast stability, models like `MQCNN`, `MQT`, and `SPADE` employ a little-known but highly effective technique: forking-sequences. Unlike standard statistical and neural forecasting methods that treat each FCD independently, the forking-sequences method jointly encodes and decodes the entire time series across all FCDs, in a way mirroring time series cross-validation. Since forking sequences remains largely unknown in the broader neural forecasting community, in this work, we formalize the forking-sequences approach, and we make a case for its broader adoption. We demonstrate three key benefits of forking-sequences: (i) more stable and consistent gradient updates during training; (ii) reduced forecast variance through ensembling; and (iii) improved inference computational efficiency. We validate forking-sequences' benefits using 16 datasets from the M1, M3, M4, and Tourism competitions, showing improvements in forecast percentage change stability of 28.8%, 28.8%, 37.9%, and 31.3%, and 8.8%, on average, for `MLP`, `RNN`, `LSTM`, `CNN`, and `Transformer`-based architectures, respectively.

1 Introduction

Forecasting plays a critical role in a wide range of domains, including energy systems, finance, economics, and healthcare analytics, where it serves as a foundation for predicting uncertain events and enabling informed decision-making production scheduling, resource allocation, planning, and long-term strategic initiatives. As operational systems become increasingly forecast-dependent, the forecasts' stability becomes just as important as their accuracy, since frequent or erratic revisions can undermine users' trust and complicate planning. Moreover, excessive forecast revisions may signal problems in the forecasting method, as the ideal forecast evolution should be restricted to incorporating new information, making necessary revisions minimal or even unpredictable [20, 46, 15]. Despite this, most research in neural forecasting has focused almost exclusively on improving predictive accuracy, overlooking the stability of forecast revisions. To be effective in real-world applications, neural forecasting methods must be designed not only to be accurate, but also to account for the evolution of forecasts over time. This includes both monitoring accuracy across forecast creation dates (FCDs) and actively promoting forecast stability. In this paper, we formally introduce the *forking-sequences* technique, a training and inference scheme that enables forecasting models to efficiently generate predictions across all FCDs. Forking-sequences acts as an architecture structural enhancement to neural forecasting models, that implicitly acts as a data augmentation technique during training,

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: GPU-Accelerated and Scalable Optimization (ScaleOpt).



(a) Original Forecasts **(b)** FS Ensembled Forecasts

Figure 1: Comparison of forecast distributions on a series from the `M1` dataset [31]. a) Forecasts generated without the forking-sequences ensemble. b) Forecasts with the forking-sequences ensemble applied. Augmenting the model with the forking-sequences ensemble significantly reduces forecast variability across forecast creation dates, resulting in more stable and consistent forecast distributions. The lines show P50 (median) forecasts across different forecast creation dates. By reusing encoder computations, forking-sequences enables computationally efficient ensembling with negligible additional computational cost.

stabilizing the gradient optimization, enables forecasting ensembles to reduce variance, and enables highly efficient temporal cross-validation inference. We discuss forking-sequences in section 2 and related work in Appendix A. Our key contributions are summarized below:

(i) **Forking-Sequences Theoretical Foundations.** We formally introduce forking-sequences and analyze its variance reduction properties. We show that, under mild temporal correlation assumptions [43], the forecast and the gradient variance decrease at a linear rate with the number of FCDs ($\mathcal{O}(1/T)$), echoing results from the weak law of large numbers. This reduction in gradient variance leads to more stable and informative updates during training, which in turn accelerates convergence in optimization, both in convex and nonconvex loss landscapes.

(ii) **Forking-Sequences Cross-Validation Computational Complexity.** We show that forking-sequences enables a new computational regime for encoder-decoder architectures by supporting efficient cross-validation-style inference. Unlike standard approaches that recompute the encoder for each forecast creation date (FCD), forking-sequences reuses encoder outputs across all FCDs, significantly reducing redundant computation. This results in orders-of-magnitude gains in inference efficiency, from quadratic to linear complexity ($\mathcal{O}(T^2)$ vs $\mathcal{O}(T)$).

(iii) **Forking-Sequences Empirical Validation.** We compare forking-sequences with the standard window-sampling approach on 16 large scale `Tourism` [3] M forecast competition datasets [31, 32, 33] in terms of: accuracy, forecast variance, and computational speed. The accuracy of various encoder models is improved by up to 43.2% and the percentage change in the forecast is reduced by up to 37.9%, on average across datasets.

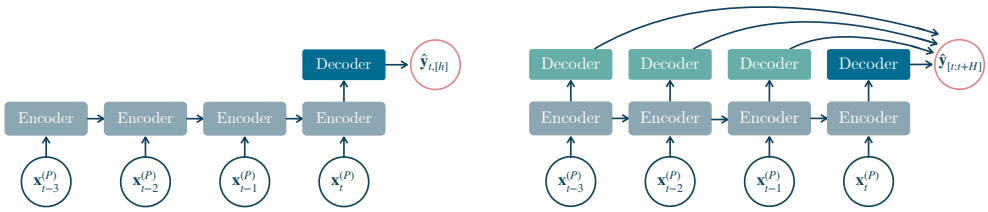
2 Methods

Here we introduce the general forecasting task notation [48, 37, 14, 49]. Let the forecast creation dates be denoted by $[:t] = [1, ..., T]$ and the forecast horizon dates be denoted by $[h] = [1, 2, ..., H]$. We consider past autoregressive features, known future information and static data, respectively, denoted $\mathbf{X}_{[:t][h]} = [\mathbf{x}^{(p)}_{[:t]},\ \mathbf{x}^{(f)}_{[t][h]},\ \mathbf{x}^{(s)}]$, and the target variable of the time series $\mathbf{Y}_{[t][h]}$. The forecasting task estimates the following conditional probability:

$$\mathbb{P}\left(\mathbf{Y}_{[t][h]} \mid \boldsymbol{\theta},\ \mathbf{X}_{[:t][h]}\right). \tag{1}$$

Model Estimation. We train models by minimizing the Quantile Loss (QL; [27]). Specifically, we optimize the model parameters $\boldsymbol{\theta}$ across nine quantiles $q \in \{0.1, 0.2, \ldots, 0.9\}$, as follows:

$$\hat{\boldsymbol{\theta}} := \underset{\boldsymbol{\theta} \in \Theta}{\arg\min}\, \mathbb{E}\left[\mathrm{QL}^{(q)}(\mathbf{y}, \hat{\mathbf{y}}^{(q)}(\boldsymbol{\theta}))\right]. \tag{2}$$



(a) Standard Window-Sampling **(b)** Forking-Sequences

Figure 2: a) A neural forecasting model trained using the *window-sampling* training scheme. Blue rectangles indicate encoder windows, while the green rectangles denotes decoder, the red circle denotes the prediction target. b) A neural forecasting model trained using the forking-sequences training-scheme. Forking-sequences collects multi-horizon errors from all intermediate forecast creation times. Additionally decoded predictions across FCDs can be easily ensembled without incurring in additional encoding computational costs.

2.1 Forking-Sequences

The standard *window-sampling* scheme is how most neural forecasting models operate [39, 9, 35, 41, 29, 13, 52, 2, 50, 16]. The approach segments the series into windows of size L. Windows are treated independently across FCDs. Figure 2a depicts the approach and Equation (3) formalizes.

$$\mathbf{h}_{t,[h]} = \text{Encoder}\left(\mathbf{X}_{[t-L:t][h]}\right) \qquad \text{and} \qquad \hat{\mathbf{y}}_{t,[h]} = \text{Decoder}\left(\mathbf{h}_{t,[h]}\right) \tag{3}$$

In contrast, the forking-sequences scheme leverages all forecast creation dates (FCDs) within a time series by reusing encoder computations, as illustrated in Figure 2b and formalized in Equation (4). When augmenting a neural forecasting architecture, forking-sequences allow the model to jointly encode the input series and then decode forecasts for all FCDs in parallel. Specifically, the model computes once a hidden representation $\mathbf{h}_{[t][h]}$, and decodes forecasts $\hat{\mathbf{y}}_{[t][h]}$ for each FCD.

$$\mathbf{h}_{[t][h]} = \text{Encoder}\left(\mathbf{X}_{[t][h]}\right) \qquad \text{and} \qquad \hat{\mathbf{y}}_{[t][h]} = \text{Decoders}\left(\mathbf{h}_{[t][h]}\right) \tag{4}$$

2.2 Forking-Sequences Training

During training, instead of sampling individual forecast creation dates t, forking-sequences gathers multi-horizon losses across all FCDs. This distinction between the forking-sequences and window-sampling training schemes can be formalized in terms of the choice of the set of FCDs over which the training loss and therefore the gradient are calculated:

$$\nabla \mathcal{L}_T = \frac{1}{|\mathcal{B}| \times |T| \times H} \sum_{b \in \mathcal{B}} \sum_{t \in T} \sum_{h=1}^{H} \nabla \text{QL}^{(q)} \left(\mathbf{y}_{b,t,h},\ \hat{\mathbf{y}}^{(q)}_{b,t,h}\right) \tag{5}$$

Theorem 1. (*Forking-Sequences Gradient Variance Reduction*) Consider the forking-sequences gradient estimator from Equation (5). If the sequence of gradient samples T is M-dependent [43] - as is typical when using correlated signals across the FCDs - then the estimator converges in probability to the full gradient, and its variance decreases at linear rate $\mathcal{O}\left(1/|T|\right)$. Full proof in Appendix D.

We show that the forking-sequences training scheme enables faster convergence in loss on the train set and reduces the variance of the stochastic gradient around its mean as shown in Fig. 6. To ensure convergence to a shared global minimizer, we train linear autoregressive models using convex quantile loss on the M1 Monthly dataset. Details on the ablation study can be found in Appendix E.

2.3 Forking-Sequences Computational Complexity

As shown in the Figure 3, the forking-sequences scheme significantly reduces the number of operations required to generate forecasts across forecast creation dates (FCDs) during cross-validation. Forking-sequences is most effective with architectures like convolutions, recurrent networks, or self-attention, which process the full sequence in a single pass, as it reuses of intermediate computations across forecast creation dates (FCDs), yielding a computational gain of factor T, where T is the number of FCDs. In the case of convolutional models, forking-sequences scales linearly with the time series length, achieving a complexity of $\mathcal{O}(T)$ by producing forecasts at each FCD while reusing previous computations.

In contrast, window-sampling incurs a higher computational cost—scaling quadratically as $\mathcal{O}(T^2)$ in the WS (full) case, or linearly as $\mathcal{O}(TL)$ when restricting the encoder to a window of length L in the WS (restricted) case. While one might argue that the gradient variance reduction shown in **Theorem 1** could also be achieved by simply increasing the batch size, forking-sequences achieves this improvement with significantly lower computational cost, as shown in Figure 3. Specifically, increasing the batch size by a factor of T would result in a computational cost of $\mathcal{O}(BT) = \mathcal{O}(T^2)$, assuming $B = T$. In contrast, forking-sequences achieves comparable variance reduction with only $\mathcal{O}(T)$ complexity, by reusing computations across FCDs.

	Encoder Complexity			
	Conv	RNN	Attention	MLPs
FS	O(T)	O(T)	O(T^2)	O(T L)
WS (restricted)	O(T L)	O(T L)	O(T^2 L)	O(T L)
WS (full)	O(T^2)	O(T^2)	O(T^3)	O(T^2)

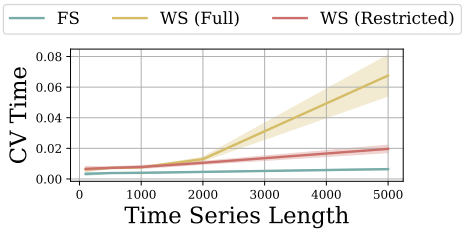


Figure 3: Computational complexity of temporal cross-validation inference methods. Here, T represents the length of the time series, and L denotes the window size in restricted window sampling.

2.4 Forking-Sequences Ensembling

Forking-sequences can be efficiently extended during inference into an ensembling technique aimed at reducing forecast variance. For each target date τ, this involves averaging predictions generated across multiple forecast creation dates, as follows:

$$\hat{\mathbf{y}}^{(q)}_{\tau,\eta} = \frac{1}{|\mathcal{H}|} \sum_{(t,h)\in\mathcal{H}} \hat{\mathbf{y}}^{(q)}_{t,h} \tag{6}$$

$$\text{where} \qquad \mathcal{H} = \mathcal{H}(\tau,\eta) = \{(t,h) \mid t+h=\tau, \text{ and } h \geq \eta\} \tag{7}$$

denotes the set of available η-step ahead forecasts for the target date τ. The ensembling procedure and the corresponding $\mathcal{H}$ set of available forecasts set are illustrated in Figure 8a.

Theorem 2. (*Forking-Sequences Forecast Variance Reduction*) Consider the ensembled forking-sequences forecasts from Equation (6). If the available of forecasts $\mathcal{H}$ for a target date τ and horizon η are unbiased and M-dependent [43], then the forecast ensemble converges in probability to the true value, and its variance decreases at rate $\mathcal{O}\left(1/|\mathcal{H}|\right)$. The proof is analogous to **Theorem 1**.

$$\hat{\mathbf{y}}^{(q)}_{\tau,\eta} = f\left(\hat{\mathbf{y}}^{(q)}_{t,h}\right) \quad \text{with} \quad (t,h) \in \mathcal{H} \tag{8}$$

3 Experiments

Datasets. To measure the effects of different training schemes, we use 16 large-scale forecasting benchmarks containing over 100,000 time series, drawn from well-known forecasting competitions: `M1` [31], `M3` [32], `M4` [33], and `Tourism` [3].

Table 3 outlines the datasets, with additional information in Appendix B. We adopt the data handling and pre-processing practices established in prior work on cross-frequency transfer learning [1, 38], more details in Appendix C.

Baselines. For our main experiments, we evaluated a curated set of baseline models. These include the classical statistical model `AutoARIMA` [25, 23], as well as neural forecasting models designed for controlled comparisons. Specifically, we fix the overall `MQForecaster` [48, 37] architecture and vary only the encoder and training scheme, considering `MLP`, `RNN`, `LSTM`, `CNN`, and `SelfAttention`. We employ and adapt the dilated `RNN` implementation from [10, 36] in our work. The `Transformer` (transf.) encoder leverages skip connections and residual layers with dilated `SelfAttention` layers inspired by [47] dilated convolutions. For each encoder-variant of `MQForecaster`, we train models with window-sampling and compare the generalization performance against that of models trained with forking-sequences. The forking-sequences models also compute a moving average forecast ensemble during inference. We ablate other ensembling techniques in Appendix G.

Table 1: Empirical evaluation of probabilistic forecasts. Mean *scaled continuous ranked probability score* (sCRPS) averaged over 5 runs. Lower measurements are preferred. The methods without standard deviation have deterministic solutions. For the `MQForecaster` architecture we vary the type of encoder, and the training scheme between *forking-sequences* (FS) and *window-sampling* (WS).

		`MLP`		`RNN`		`LSTM`		`CNN`		Transf.		StatsForecast	
	Freq	FS	WS	FS	WS	FS	WS	FS	WS	FS	WS	`ETS`	`ARIMA`
M1	M	0.13 (0.0034)	0.1677 (0.0005)	0.1283 (0.0021)	0.1678 (0.0006)	0.1295 (0.0039)	0.2124 (0.0165)	0.1286 (0.0005)	0.1698 (0.0009)	0.1603 (0.0012)	0.168 (0.0012)	0.1418 (-)	0.1509 (-)
	Q	0.1118 (0.002)	0.1154 (0.0003)	0.1082 (0.0015)	0.1151 (0.0023)	0.1088 (0.0018)	0.1331 (0.0126)	0.1117 (0.0009)	0.1137 (0.0049)	0.1125 (0.0013)	0.1162 (0.0018)	0.1139 (-)	0.1282 (-)
	Y	0.0975 (0.002)	0.0953 (0.0006)	0.102 (0.0017)	0.0968 (0.0016)	0.1005 (0.0029)	0.2188 (0.0527)	0.1049 (0.0019)	0.1068 (0.0053)	0.1059 (0.003)	0.0938 (0.002)	0.1097 (-)	0.1068 (-)
M3	O	0.0365 (0.0001)	0.037 (0.0001)	0.0361 (0.0002)	0.0378 (0.0005)	0.0384 (0.0011)	0.0891 (0.0091)	0.0374 (0.0004)	0.0393 (0.0021)	0.0361 (0.0005)	0.0372 (0.0001)	0.0328 (-)	0.0337 (-)
	M	0.0958 (0.0027)	0.1122 (0.0004)	0.0915 (0.0013)	0.1133 (0.0003)	0.0917 (0.0012)	0.1407 (0.0136)	0.0925 (0.0001)	0.1137 (0.0006)	0.1106 (0.0013)	0.1129 (0.0008)	0.105 (-)	0.1059 (-)
	Q	0.0778 (0.0001)	0.0861 (0.0004)	0.0757 (0.0005)	0.0866 (0.0007)	0.0758 (0.0006)	0.1173 (0.0083)	0.0757 (0.0003)	0.0888 (0.0013)	0.0847 (0.001)	0.0864 (0.0004)	0.0773 (-)	0.0779 (-)
	Y	0.1554 (0.0032)	0.1689 (0.0004)	0.1517 (0.0004)	0.1696 (0.0023)	0.15 (0.0003)	0.3352 (0.0741)	0.1522 (0.0003)	0.1672 (0.0014)	0.1667 (0.0068)	0.1687 (0.0009)	0.149 (-)	0.1549 (-)
M4	H	0.0309 (0.0015)	0.0873 (0.0016)	0.0327 (0.0028)	0.0879 (0.0021)	0.0378 (0.007)	0.1067 (0.0035)	0.0322 (0.001)	0.0903 (0.0019)	0.0815 (0.0008)	0.0907 (0.002)	0.0684 (-)	0.0308 (-)
	D	0.0237 (0)	0.024 (0.0001)	0.0242 (0.0001)	0.0253 (0.0001)	0.0242 (0.0003)	0.0407 (0.0067)	0.0239 (0.0001)	0.0282 (0.0021)	0.024 (0)	0.024 (0.0001)	0.0226 (-)	0.0226 (-)
	W	0.0502 (0.0009)	0.0558 (0.0001)	0.0479 (0.0024)	0.057 (0.0007)	0.0492 (0.0035)	0.0798 (0.007)	0.0444 (0.0006)	0.0576 (0.0005)	0.0544 (0.0005)	0.0563 (0.0003)	0.0542 (-)	0.0507 (-)
	M	0.0966 (0.0024)	0.1063 (0.0005)	0.093 (0.0005)	0.1075 (0.0004)	0.093 (0.0012)	0.1366 (0.0122)	0.0931 (0.0003)	0.1086 (0.001)	0.1049 (0.0005)	0.1067 (0.0003)	0.097 (-)	0.0987 (-)
	Q	0.0871 (0.0001)	0.0946 (0.0005)	0.0853 (0.0002)	0.0953 (0.001)	0.0845 (0.0002)	0.1248 (0.0079)	0.0852 (0.0003)	0.0991 (0.0017)	0.0931 (0.0013)	0.0949 (0.0006)	0.0807 (-)	0.0849 (-)
	Y	0.1293 (0.003)	0.1365 (0.0004)	0.1264 (0.0002)	0.1372 (0.0014)	0.1257 (0.0001)	0.2813 (0.0655)	0.1268 (0.0002)	0.1425 (0.0028)	0.1363 (0.0066)	0.1366 (0.0002)	0.124 (-)	0.1322 (-)
Tourism	M	0.099 (0.0121)	0.2183 (0.001)	0.0871 (0.0039)	0.2196 (0.002)	0.0936 (0.0104)	0.2687 (0.0257)	0.0927 (0.0018)	0.2195 (0.002)	0.2053 (0.0047)	0.2207 (0.0026)	0.1198 (-)	0.1479 (-)
	Q	0.0967 (0.0008)	0.1238 (0.0021)	0.0963 (0.0005)	0.1246 (0.0016)	0.0951 (0.001)	0.1545 (0.009)	0.0929 (0.0006)	0.1282 (0.0031)	0.1243 (0.0044)	0.1245 (0.0013)	0.1042 (-)	0.1187 (-)
	Y	0.1891 (0.0064)	0.1825 (0.0004)	0.1822 (0.0015)	0.1841 (0.0006)	0.1831 (0.0013)	0.264 (0.0343)	0.1828 (0.0011)	0.1934 (0.0059)	0.1807 (0.0031)	0.1828 (0.0002)	0.1413 (-)	0.1697 (-)

3.1 Probabilistic Forecast Accuracy Results

We measure the *scaled Continuous Ranked Probability Score* (sCRPS, [18]), as defined in Equation (9). The sCRPS is a scaled version of the CRPS.

$$\mathrm{sCRPS}\left(\mathbf{y}_{[b][t][h]},\ \hat{\mathbf{Y}}_{[b][t][h]}\right) = \frac{\sum_{b,t,h} \mathrm{CRPS}(y_{b,t,h}, \hat{Y}_{b,t,h})}{\sum_{b,t,h} |y_{b,t,h}|}. \tag{9}$$

Table 1 reports mean sCRPS values, averaged over five runs, for all dataset–frequency combinations and model architectures, including statistical baselines. For `LSTM` encoder models, the forking-sequences training scheme reduced sCRPS by 43.2%, on average across datasets, compared to the window-sampling scheme, confirming earlier observations by [30, 12, 28]. Consistent gains are observed for `MLP` (14.3%), `RNN` (16.9%), and `CNN` (19.2%) encoders, while the `Transformer`-based encoder demonstrated relatively less improvement (1.8%), likely due to its superior gradient flow reducing information bottlenecks. The results of Table 1 are summarized in Fig. 10c which shows the percentage change of the sCRPS metric for models with the forking-sequences training scheme with ensembling applied during inference, compared to models using the window-sampling scheme, averaged across datasets. All encoder variants with forking-sequences have improved sCRPS and generally outperform statistical baselines. Point forecasting results are reported in Appendix H.

3.2 Forecast Revision Results

We evaluate forecast revisions using the *Symmetric Quantile Percentage Change* (sQPC), that we define in Equation (10). sQPC measures the relative change in predicted quantiles across consecutive forecast creation dates, providing a quantitative view of temporal instability or forecast revision rates. Inspired by the sMAPE metric, sQPC uses a symmetric denominator, based on both current and previous forecasts, to mitigate issues of numerical instability [26]. This design ensures robustness

Table 2: Empirical evaluation of probabilistic forecasts. Mean *symmetric quantile percentage change* (sQPC) averaged over 5 runs. Lower measurements are preferred. The methods without standard deviation have deterministic solutions. For the `MQForecaster` architecture we vary the type of encoder, and the training scheme between *forking-sequences* (FS) and *window-sampling* (WS).

		`MLP`		`RNN`		`LSTM`		`CNN`		Transf.		StatsForecast	
	Freq	FS	WS	FS	WS	FS	WS	FS	WS	FS	WS	`ETS`	`ARIMA`
M1	M	1.0462 (0.0264)	1.8328 (0.112)	0.9907 (0.0185)	1.753 (0.0445)	0.9738 (0.0341)	1.4111 (0.4895)	1.0057 (0.0242)	1.8798 (0.03)	1.56 (0.0712)	1.859 (0.1749)	2.0154 (-)	2.5845 (-)
	Q	1.2945 (0.013)	1.872 (0.0316)	1.3491 (0.0269)	1.8102 (0.0721)	1.3113 (0.0137)	1.2636 (0.3054)	1.3058 (0.0217)	1.7992 (0.0867)	1.6754 (0.0765)	1.865 (0.0584)	3.6355 (-)	5.6424 (-)
	Y	2.1431 (0.0857)	2.1515 (0.0399)	1.9734 (0.0152)	2.1159 (0.0324)	1.905 (0.014)	2.4131 (0.5608)	1.9832 (0.0186)	2.0665 (0.0623)	1.9857 (0.0767)	2.1233 (0.0504)	2.8426 (-)	3.4499 (-)
M3	O	0.2636 (0.0033)	0.3799 (0.0083)	0.2643 (0.0087)	0.3825 (0.0071)	0.2564 (0.0162)	0.5743 (0.0673)	0.2617 (0.0054)	0.3876 (0.0468)	0.2981 (0.0203)	0.3984 (0.0138)	0.8979 (-)	0.8581 (-)
	M	0.6673 (0.0151)	1.2841 (0.0684)	0.5709 (0.0107)	1.2568 (0.0344)	0.5579 (0.0112)	1.1896 (0.4474)	0.5986 (0.0031)	1.3169 (0.0286)	1.1019 (0.0758)	1.3176 (0.1119)	1.3749 (-)	1.7204 (-)
	Q	0.6065 (0.0071)	0.8967 (0.0184)	0.6067 (0.0077)	0.8633 (0.0337)	0.5919 (0.0016)	0.7163 (0.1731)	0.5881 (0.0062)	0.8469 (0.0536)	0.8047 (0.0389)	0.8859 (0.0277)	1.8586 (-)	2.1937 (-)
	Y	1.5485 (0.0822)	1.7565 (0.0179)	1.5048 (0.0105)	1.6539 (0.0142)	1.4681 (0.0104)	2.1152 (0.6829)	1.4833 (0.012)	1.7026 (0.0693)	1.5973 (0.0263)	1.7463 (0.0377)	3.4889 (-)	5.0013 (-)
M4	H	0.7967 (0.0946)	2.5037 (0.2128)	0.8332 (0.1019)	2.5784 (0.1889)	0.5486 (0.0807)	5.1758 (0.752)	0.6432 (0.0854)	2.0002 (0.2027)	2.5235 (0.2192)	2.1408 (0.2919)	2.4881 (-)	0.7861 (-)
	D	0.1611 (0.0005)	0.1652 (0.0017)	0.1749 (0.0015)	0.1601 (0.0018)	0.1533 (0.0051)	0.3365 (0.0388)	0.1667 (0.0009)	0.1751 (0.012)	0.1639 (0.0005)	0.1636 (0.0047)	0.497 (-)	0.4992 (-)
	W	0.4577 (0.0336)	0.4903 (0.0094)	0.3928 (0.0351)	0.478 (0.0052)	0.3165 (0.0442)	1.097 (0.3924)	0.3645 (0.024)	0.4755 (0.0343)	0.4626 (0.0462)	0.4849 (0.0067)	1.4943 (-)	1.1399 (-)
	M	0.6812 (0.014)	0.9514 (0.0445)	0.6506 (0.0129)	0.9094 (0.0181)	0.6478 (0.0038)	1.2143 (0.4526)	0.6661 (0.0057)	0.9818 (0.0237)	0.8547 (0.0237)	0.9625 (0.0738)	1.8584 (-)	2.2324 (-)
	Q	0.6333 (0.0043)	0.8047 (0.0094)	0.6199 (0.0083)	0.7882 (0.0328)	0.6035 (0.0047)	0.8546 (0.2459)	0.6129 (0.0035)	0.7798 (0.0513)	0.7391 (0.0242)	0.8066 (0.029)	2.0469 (-)	2.2461 (-)
	Y	1.1631 (0.0517)	1.167 (0.0147)	1.112 (0.0074)	1.1227 (0.0087)	1.0829 (0.0033)	1.6507 (0.4816)	1.0847 (0.0074)	1.1098 (0.0429)	1.1059 (0.0155)	1.1596 (0.0151)	3.082 (-)	4.5323 (-)
Tourism	M	0.9024 (0.0414)	3.3663 (0.1779)	0.9205 (0.0368)	3.2636 (0.1442)	0.8345 (0.0412)	2.1799 (1.0244)	0.8876 (0.0221)	3.4973 (0.0846)	2.8324 (0.2065)	3.5895 (0.3679)	2.1456 (-)	1.7748 (-)
	Q	0.9548 (0.0086)	2.7272 (0.0399)	0.963 (0.0363)	2.6067 (0.1216)	0.9271 (0.0158)	1.4409 (0.5111)	0.8857 (0.0107)	2.7894 (0.1924)	2.3121 (0.1814)	2.7418 (0.14)	3.1549 (-)	2.7744 (-)
	Y	3.4661 (0.0416)	3.4306 (0.0461)	3.3787 (0.0354)	3.343 (0.0671)	3.3273 (0.0526)	2.7505 (0.3021)	3.3623 (0.0166)	3.29 (0.1324)	3.3494 (0.0652)	3.4032 (0.0663)	9.7853 (-)	12.016 (-)

when dealing with small predicted values and avoids the division-by-zero problems common in traditional percentage-based metrics.

$$\mathrm{sQPC}_q\left(\hat{\mathbf{Y}}^{(q)}_{[b][t][h]}\right) = \frac{200}{B \times T \times H} \sum_{b,t,h} \frac{|\hat{Y}^{(q)}_{b,t+1,h} - \hat{Y}^{(q)}_{b,t,h+1}|}{|\hat{Y}^{(q)}_{b,t+1,h}| + |\hat{Y}^{(q)}_{b,t,h+1}|} \tag{10}$$

Because uncertainty decreases across FCDs it is expected that forecasted quantiles above P50 decrease while below P50 increase, to avoid complications, we measure the revisions for the median (q=.50).

Table 2 presents sQPC values, averaged over five runs, for all dataset–frequency combinations and model architectures, including statistical baselines. For `LSTM` encoder models, the forking-sequences scheme reduces forecast revisions by 37.9%, on average across datasets, compared to window-sampling.

Consistent gains are observed for `MLP` (28.8%), `RNN` (28.8%), and `CNN` (31.3%) encoders, while the `Transformer`-based encoder has relatively less improvement (8.8%). Table 2 is summarized in Fig. 10d, which shows the percentage change of the sQPC metric for models using the *forking-sequences* training scheme with ensembling applied during inference, compared to models using the *window-sampling* scheme, averaged across datasets. All encoder variants with forking-sequences show improved sQPC. Ensembling during inference results in a marginal reduction in sCRPS across models, with substantial improvements in sQPC compared to no ensembling during inference, as shown in Figs. 10a and 10b.

4 Discussion and Conclusion

In this work we skipped almost all hyperparameter tuning, keeping it minimal. In future work we will explore whether more nuanced hyperparameter selection for models with the window-sampling training scheme can help bridge the error gap with forking-sequences. Additionally, "smarter" sampling strategies could select more informative windows: while our current approach avoids windows with missing or padded targets, it still selects randomly. Alternatives include sampling a fixed number of windows or increasing window size during training via a step-size hyperparameter.

Through theoretical analysis and extensive empirical evaluation on 16 major forecasting benchmarks, from the M1, M3, M4 and Tourism competitions, we demonstrated three main benefits from forking-sequences: (1) reduced gradient variance during training, leading to faster model optimization convergence; (2) computationally efficient cross-validation inference via encoder computation reuse, which is orders of magnitude faster than windows-sampling approach; and (3) lower forecast variance through ensembling. Forking-sequences has yet to gain widespread adoption due in large part to the dominant trend of simple window-sampling architectures reinforced by the major neural forecasting libraries [51], [21], [1], [5], and [36]. We hope this study motivates maintainers of these libraries to adopt or re-introduce [36] forking-sequences.

Acknowledgments

The authors wish to thank Ruijun Ma for his guidance and informative conversations.

References

[1] Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Tarkmen, and Yuyang Wang. GluonTS: Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020.

[2] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series, 2024.

[3] George Athanasopoulos, Rob J. Hyndman, Haiyan Song, and Doris C. Wu. The Tourism forecasting competition. *International Journal of Forecasting*, 27(3):822–844, 2011. Special Section 1: Forecasting with Artificial Neural Networks and Computational Intelligence. Special Section 2: Tourism Forecasting.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015*, 2016.

[5] Jan Beitner. PyTorchForecasting: Forecasting with neural networks made simple. GitHub Repository, 2020.

[6] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

[7] Christoph Bergmeir and José M Benítez. On the use of cross-validation for time series predictor evaluation. *Information Sciences*, 191:192–213, 2012.

[8] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle,

M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[9] Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler-Canseco, and Artur Dubrawski. NHITS: neural hierarchical interpolation for time series forecasting. In *Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'23/IAAI'23/EAAI'23. AAAI Press, 2023.

[10] Shiyu Chang, Yang Zhang, Wei Han, Mo Yu, Xiaoxiao Guo, Wei Tan, and et al. Dilated recurrent neural networks. In *31st Conference on Neural Information Processing Systems*, 2017.

[11] Kevin C. Chen, Lee Dicker, Carson Eisenach, and Dhruv Madeka. MQTransformer: Multi-Horizon Forecasts with Context Dependent Attention and Optimal Bregman Volatility. In Maria Florina Balcan and Marina Meila, editors, *In: Proceedings of 8th SIGKDD International Workshop on Mining and Learning From Time Series - Deep Forecasting: Models, Interpretability, and Applications (KDD' 22)*. ACM. Association for Computing Machinery, 8 2022.

[12] Andrew M. Dai and Quoc V. Le. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems, NIPS*, 2015.

[13] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting, 2024.

[14] Carson Eisenach, Yagna Patel, and Dhruv Madeka. MQTransformer: Multi-Horizon Forecasts with Context Dependent and Feedback-Aware Attention. In Maria Florina Balcan and Marina Meila, editors, *Submitted to Proceedings of the 38th International Conference on Machine Learning*. PMLR. Working Paper version available at arXiv:2009.14799, 8 2021.

[15] Dean P. Foster and Robert A. Stine. Threshold martingales and the evolution of forecasts, 2021.

[16] Azul Garza and Max Mergenthaler-Canseco. Timegpt, 2023.

[17] Federico Garza, Max Mergenthaler Canseco, Cristian Challú, and Kin G. Olivares. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022.

[18] Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007.

[19] Jeffrey D. Hart. Automated kernel smoothing of dependent data by using time series cross-validation. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(3):529–542, 1994.

[20] DAVID C. HEATH and PETER L. JACKSON. Modeling the evolution of demand forecasts ith application to safety stock analysis in production/distribution systems. *IIE Transactions*, 26(3):17–30, 1994.

[21] Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅ›cisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and GaÃ«l Grosch. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6, 2022.

[22] Charles C Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *(O.N.R. Memorandum No. 52)*, 1957.

[23] Rob J Hyndman, George Athanasopoulos, Azul Garza, Cristian Challu, Max Mergenthaler, and Kin G. Olivares. *Forecasting: Principles and Practice, the Pythonic Way*. OTexts, Melbourne, Australia, 2025. available at https://otexts.com/fpppy/.

[24] Rob J. Hyndman and Baki Billah. Unmasking the theta method. *International Journal of Forecasting*, 19(2):287–290, 2003.

[25] Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software, Articles*, 27(3):1–22, 2008.

[26] Rob J. Hyndman and Anne B. Koehler. Another look at measures of forecast accuracy. *International Journal of Forecasting*, 22(4):679 – 688, 2006.

[27] Roger Koenker and Gilbert Bassett. Regression quantiles. *Econometrica*, 46(1):33–50, 1978.

[28] Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-Supervised Nets. In Guy Lebanon and S. V. N. Vishwanathan, editors, *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, volume 38 of *Proceedings of Machine Learning Research*, pages 562–570, San Diego, California, USA, 09–12 May 2015. PMLR.

[29] 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.

[30] Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to diagnose with LSTM recurrent neural networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[31] S. Makridakis, A. Andersen, R. Carbone, R. Fildes, M. Hibon, R. Lewandowski, J. Newton, E. Parzen, and R. Winkler. The accuracy of extrapolation (time series) methods: Results of a forecasting competition. *Journal of Forecasting*, 1(2):111–153, 1982.

[32] Spyros Makridakis and Michèle Hibon. The M3-competition: results, conclusions and implications. *International Journal of Forecasting*, 16(4):451–476, 2000. The M3- Competition.

[33] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1):54–74, 2020. M4 Competition.

[34] Albert W. Marshall and Ingram Olkin. Multivariate Chebyshev Inequalities. *The Annals of Mathematical Statistics*, 31(4):1001 – 1014, 1960.

[35] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *International Conference on Learning Representations*, 2023.

[36] Kin G. Olivares, Cristian Challú, Federico Garza, Max Mergenthaler Canseco, and Artur Dubrawski. NeuralForecast: User friendly state-of-the-art neural forecasting models. PyCon Salt Lake City, Utah, US 2022, 2022.

[37] Kin G. Olivares, Nganba Meetei, Ruijun Ma, Rohan Reddy, Mengfei Cao, and Lee Dicker. Probabilistic hierarchical forecasting with deep poisson mixtures. *International Journal of Forecasting, accepted*, Preprint version available at arXiv:2110.13179, 2023.

[38] Kin G. Olivares, Malcolm Wolff, Tatiana Konstantinova, Shankar Ramasubramanian, Boris Oreshkin, Andrew Gordon Wilson, Andres Potapczynski, Willa Potosnak, Mengfei Cao, Michael W. Mahoney, and Dmitry Efimov. A realistic evaluation of cross-frequency transfer learning and foundation forecasting models. In *Thirty-Ninth Annual Conference on Neural Information Processing Systems NeurIPS 2025*, volume Recent Advances in Time Series Foundation Models Have We Reached the 'BERT Moment'?, San Diego, USA, 2025. NeurIPS 2025.

[39] Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

[40] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. `https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf`, 2018.

[41] 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.

[42] Artemios-Anargyros Semenoglou, Evangelos Spiliotis, Spyros Makridakis, and Vassilios Assimakopoulos. Investigating the accuracy of cross-learning time series forecasting methods. *International Journal of Forecasting*, 37(3):1072–1084, 2021.

[43] R.H. Shumway and D.S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer Texts in Statistics. Springer International Publishing, 2017.

[44] Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 07 2019.

[45] M Syntetos, John Boylan, and JD Croston. On the categorization of demand patterns. *Journal of the Operational Research Society*, 56, 05 2005.

[46] L. Beril Toktay and Lawrence M. Wein. Analysis of a forecasting-production-inventory system with stationary demand. *Management Science*, 47(9):1268–1281, 2001.

[47] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A generative model for raw audio. *Computer Research Repository*, abs/1609.03499, 2016.

[48] Ruofeng Wen, Kari Torkkola, Balakrishnan Narayanaswamy, and Dhruv Madeka. A Multi-horizon Quantile Recurrent Forecaster. In *31st Conference on Neural Information Processing Systems NIPS 2017, Time Series Workshop*, 2017.

[49] Malcolm Wolff, Kin G. Olivares, Boris Oreshkin, Sunny Ruan, Sitan Yang, Abhinav Katoch, Shankar Ramasubramanian, Youxin Zhang, Michael W. Mahoney, Dmitry Efimov, and Vincent Quenneville-Bélair. ♠ SPADE ♠: Split Peak Attention DEcomposition. In *Thirty-Eighth Annual Conference on Neural Information Processing Systems NeurIPS 2024*, volume Time Series in the Age of Large Models Workshop, Vancouver, Canada, 2024. NeurIPS 2024.

[50] Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers, 2024.

[51] Haixu Wu, Yong Liu, Huikun Weng, Yuxuan Wang, Tengge Hu, Haoran Zhang, and Jiawei Guo. Time Series Library (TSLib). GitHub Repository, 2023.

[52] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting. *The Association for the Advancement of Artificial Intelligence Conference 2021 (AAAI 2021).*, abs/2012.07436, 2020.

A Related Work

Similar ideas to forking-sequences have been in the forecasting literature for decades. Hart [19] introduced time-series cross-validation as a model selection technique that accounts for temporal dynamics and data dependencies, addressing the limitations of classical cross-validation, which assumes independent FCD observations [7]. Figure 4 depicts the temporal cross-validation evaluation.

Forking-sequences has informed the design of `MQForecaster` neural networks for industrial applications, with notable examples including `MQCNN`, `MQT`, and `SPADE` [48, 37, 14, 49]. However, prior work largely adopted forking-sequences as a practical heuristic, without providing formal theoretical justification or systematic empirical validation of its benefits. In some cases, its theoretical motivation was misattributed to the martingale properties of ensemble updates [11, 15], rather than recognizing its more simple and natural role in reducing variance during training and inference.

Beyond forecasting and in the context of healthcare and natural language processing, Lipton introduced the concept of target replication to train LSTM time series classifiers for medical diagnosis [30], while Dai & Le applied a similar scheme to train text document classifiers [12]. Lee et al. [28] proposed having intermediate objective losses.

The vanishing gradient problem arising from backpropagation through time in long sequences also motivated the introduction of attention mechanisms [4]. Attention improved information flow by allowing the decoder to access a weighted combination of encoder hidden states. While such architectural innovations significantly enhanced gradient propagation, we show in Section 3 that the forking-sequences approach provides complementary improvements that stabilize training, enhance forecast accuracy and reduce forecast variance.

Finally, a training scheme analogous to forking-sequences has long been a cornerstone in natural language processing (NLP), most notably through the next-word prediction objective [6]. This autoregressive learning strategy, which trains models to predict the next token given all previous ones, laid the foundation for modern language models. It was later popularized at scale by generative pre-trained transformers such as GPT-1 and GPT-3 [40, 8]. Training and inference schemes related to forking-sequences have remained underexplored in neural prediction, highlighting a clear technical gap between NLP and forecasting practice that this work aims to bridge.

B Dataset Details

Table 3: Summary of forecasting datasets used in our empirical study.

	Frequency	Seasonality	Horizon	Series	Min Length	Max Length	% Erratic
M1	Monthly	12	18	617	48	150	0
	Quarterly	4	8	203	18	114	0
	Yearly	1	6	181	15	58	0
M3	Other	4	8	174	71	104	0
	Monthly	12	18	1428	66	144	2
	Quarterly	4	8	756	24	72	1
	Yearly	1	6	645	20	47	10
M4	Hourly	24	48	414	748	1008	17
	Daily	1	14	4,227	107	9933	2
	Weekly	1	13	359	93	2610	16
	Monthly	12	18	48,000	60	2812	6
	Quarterly	4	8	24,000	24	874	11
	Yearly	1	6	23,000	19	841	18
Tourism	Monthly	12	18	366	91	333	51
	Quarterly	4	8	427	30	130	39
	Yearly	1	6	518	11	47	23

B.1 M1 Dataset Details

The early M1competition [31], organized by Makridakis et al., focused on 1,001 time series drawn from demography, industry, and economics, with lengths ranging from 9 to 132 observations and varying in frequency (monthly, quarterly, and yearly). A key empirical finding of this competition was that simple forecasting methods, such as ETS [22], often outperformed more complex approaches. These results had a lasting impact on the field, initiating a research legacy that emphasized accurate forecasting, model automation, and caution against overfitting. The competition also marked a conceptual shift, helping to distinguish time-series forecasting from traditional time series analysis.

B.2 M3 Dataset Details

The M3competition [32], held two decades after the M1competition, featured a dataset of 3,003 time series spanning business, demography, finance and economics. These series ranged from 14 to 126 observations and included monthly, quarterly, and yearly frequencies. All series had positive values, with only a small proportion displaying erratic behavior and none exhibiting intermittency [45]. The M3competition reinforced the trend of simple forecasting methods outperforming more complex alternatives, with the `Theta` method [24] emerging as the best performing approach.

B.3 M4 Dataset Details

The M4competition marked a substantial increase in both the size and diversity of the M competition datasets, comprising 100,000 time series across six frequencies: hourly, daily, weekly, monthly, quarterly, and annual. These series covered a wide range of domains, including demography, finance, industry, and both micro- and macroeconomic indicators. The competition also introduced the evaluation of prediction intervals in addition to point forecasts, broadening the assessment criteria. M4's proportion of non-smooth or erratic time series increased to 18 percent [45]. For the first time, a neural forecasting model - ESRNN[44] - outperformed traditional methods. The competition also helped popularize cross-learning [42] in global models.

B.4 Tourism Dataset Details

The Tourism dataset [3] was designed to evaluate forecasting methods applied to tourism demand data across multiple temporal frequencies. It comprises 1,311 time series at monthly, quarterly, and yearly frequencies. This competition introduced the Mean Absolute Scaled Error (MASE) as an alternative metric to evaluate scaled point forecasts, alongside the evaluation of forecast intervals. Notably, 36% of the series were classified as erratic or intermittent. Due to this high proportion of irregular data, the Naïve1 method proved particularly difficult to outperform at the yearly frequency.

C Training Methodology and Hyperparameters

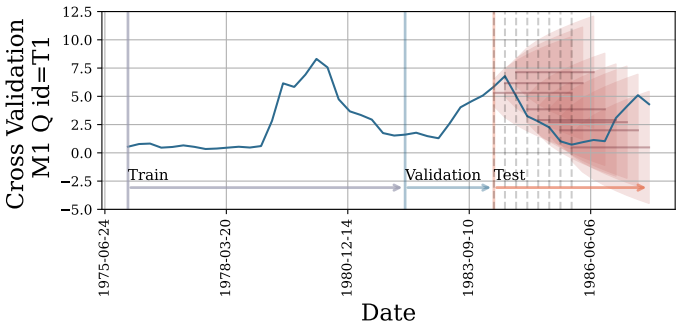


Figure 4: Example time series from the M1 competition dataset. The train and validation sets consist of all observations preceding the first dotted line. The cross validation test set is comprised of a set of forecast creation dates with lengths defined in Table 3. To leverage all the forecast creation dates within train for model estimation while preventing temporal leakage between splits, we apply masking across training and test sets..

Here, we expand on the training methodology outlined in Section 3. To align with current neural forecasting standards, we follow the data processing protocols established by `Chronos` [1, 2] and `NBEATS` [39]. We deviate slightly, by enabling a temporal cross validation evaluation with a partition of the datasets depicted in Figure 4,

We train frequency-specialized models combining M1, M3, M4and Tourismfrequency-specific datasets to leverage a larger training data corpus while still focusing the forecasting tasks to frequency-specific prediction horizons (Table 3. We default to these frequency-specific horizon values for the Tourism dataset as well for consistency across experiments. To enable temporal cross-validation in our experiments, we expand the original test sets of a single forecast horizon to include earlier timesteps for n prediction windows consecutively shifted by one time stamp. We adjust the validation set accordingly to equal a single forecast horizon preceding the test set with the train set preceding the validation set. Let h be the forecast horizon as specified in Table 3, we set $n = h$, and define the train, validation, and test partitions of the data $\mathbf{X}$ as follows.

$$\mathbf{X}_{\text{train}} = \mathbf{X}_{[\,:\,-H\times 3+1]}, \qquad \mathbf{X}_{\text{validation}} = \mathbf{X}_{[-H\times 3+1\,:\,-H\times 2+1]}, \qquad \text{and} \qquad \mathbf{X}_{\text{test}} = \mathbf{X}_{[-H\times 2+1\,:\,T]}. \tag{11}$$

We train multi-quantile loss (MQ) models varying only the encoder type to include `MLP`, `RNN`, `LSTM`, `CNN`, and `Transformer` architectures using the hyperparameters outlined in Tables 4, 5, 6, 7, 8, respectively.

We conducted all neural network experiments using a single AWS p4d.24xlarge with 1152 GiB of RAM and 96 vCPUs. Training times mostly depend on the architecture, however we restrict the SGD training steps to 30K per architectures.

C.1 MQ Forecaster Model's hyperparameters

Table 4: `MLP`

HYPERPARAMETER	VALUES
Single GPU SGD Batch Size*.	8
Initial learning rate.	0.001
Maximum Training steps S_{max}.	30,000
Learning rate decay.	0.1
Learning rate step size.	10,000
Scaler type.	Standard
Input size.	
forking-sequences	$2 * H$
window-sampling	1
Main Activation Function.	ReLU
Number of Layers.	3
Encoder Dimension.	128
H-Agnostic Decoder Dimension.	100
H-Specific Decoder Dimension.	20

Table 5: `RNN`

HYPERPARAMETER	VALUES
Single GPU SGD Batch Size*.	8
Initial learning rate.	0.001
Maximum Training steps S_{max}.	30,000
Learning rate decay.	0.1
Learning rate step size.	10,000
Scaler type.	Standard
Main Activation Function	ReLU
Encoder Dimension.	128
Number of Layers.	2
Dilations.	$[[1, 2], [4, 8]]$
H-Agnostic Decoder Dimension.	100
H-Specific Decoder Dimension.	20

Table 6: `LSTM`

HYPERPARAMETER	VALUES
Single GPU SGD Batch Size*.	8
Initial learning rate.	0.001
Maximum Training steps S_{max}.	30,000
Learning rate decay.	0.1
Learning rate step size.	10,000
Scaler type.	Standard
Main Activation Function	ReLU
Encoder Dimension.	128
Number of Layers.	2
Dilations.	$[[1, 2], [4, 8]]$
H-Agnostic Decoder Dimension.	100
H-Specific Decoder Dimension.	20

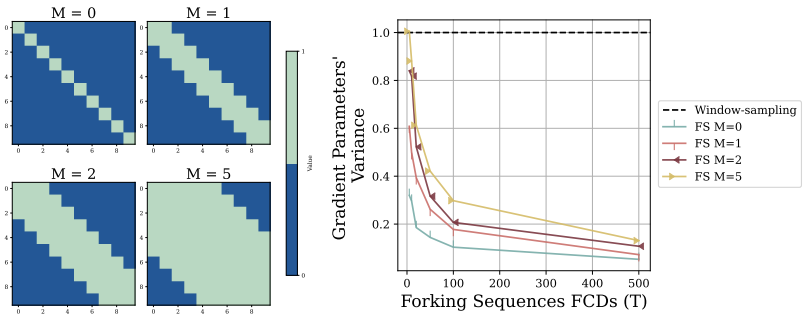
Table 7: `CNN`

HYPERPARAMETER	VALUES
Single GPU SGD Batch Size*.	8
Initial learning rate.	0.001
Maximum Training steps S_{max}.	30,000
Learning rate decay.	0.1
Learning rate step size.	10,000
Scaler type.	Standard
Main Activation Function	ReLU
Temporal Convolution Kernel Size	2
Temporal Convolution Dilations.	$[1, 2, 4, 8, 16, 32]$
H-Agnostic Decoder Dimension.	100
H-Specific Decoder Dimension.	20

Table 8: `Transformer`

HYPERPARAMETER	VALUES
Single GPU SGD Batch Size*.	8
Initial learning rate.	0.001
Maximum Training steps S_{max}.	30,000
Learning rate decay.	0.1
Learning rate step size.	10,000
Scaler type.	Standard
Encoder Hidden Size.	128
Number of Layers.	3
Patching Lengths.	$[2, 6, 8]$
Attention Dropout.	0.1
Number of Attention Heads.	4
H-Agnostic Decoder Dimension.	100
H-Specific Decoder Dimension.	20

D Forking-Sequences Theoretical Foundations



(a) M-dependence Covariance **(b)** Gradient Variance

Figure 5: a) Visualization of the covariance of M-dependent random variables. Green sections indicate correlated samples, while blue indicate uncorrelated variables. b) Mean estimator variance reduction as a function of the samples of the forking-sequences training scheme for different levels of M-dependence.

In this Appendix, we provide a proof of the gradient variance reduction guarantees for the forking-sequences training scheme introduced in Section 2. Specifically, we show that this scheme reduces the variance of the stochastic gradient around its mean, assuming short-range dependence among gradients across forecast creation dates. Our arguments build on techniques and definitions from Shumway & Stuffer [43] and Marshall & Olkin [34]. We also include a simulation (Figure 5) that empirically validates the variance reduction behavior of the gradient estimator.

For simplicity of the arguments' notation we refer to the forking-sequences gradient estimator and its samples only in terms of their FCDs $T = |\mathcal{T}|$:

$$\bar{\nabla} L_T = \frac{1}{T} \sum_t \nabla L_t$$

Definition. Let $\{\nabla L_t\} \subset \mathbb{R}^P$ be a sequence of random variables, it is M-dependent if:

- The gradient estimator is unbiased $\mathbb{E}[\nabla L_t] = \mu$,
- with covariance $\Sigma = \mathrm{Cov}(\nabla L_s, \nabla L_t) = \gamma(|s-t|) \in \mathbb{R}^{P \times P}$.
- where $\gamma(b) = \mathrm{Cov}(\nabla L_0, \nabla L_b)$ and $\gamma(b) = 0$ for all $|b| > K$

Lemma. Consider the forking sequences gradient estimator $\bar{\nabla} L_T = \frac{1}{T} \sum_t \nabla L_t$.

If the gradient samples are K dependent, the covariance of the gradient estimator almost surely converges to 0 as T grows, that is, $\mathrm{Cov}\left(\bar{\nabla} L_T\right) \xrightarrow{a.s.} 0$,

Proof.

$$\begin{aligned}
\mathrm{Cov}(\bar{\nabla} L_T) &= \frac{1}{T^2} \mathrm{Cov}\left(\sum_{t=1}^{T} \nabla L_t\right) = \frac{1}{T^2} \sum_{s=1}^{T} \sum_{t=1}^{T} \mathrm{Cov}(\nabla L_s, \nabla L_t) \\
&= \frac{1}{T^2} \sum_{b=-K}^{K} (T - |b|)\gamma(b) = \frac{1}{T} \sum_{b=-K}^{K} \left(1 - \frac{|b|}{T}\right) \gamma(b)
\end{aligned}$$

We can conclude the almost sure convergence as $\frac{|b|}{T} \to 0$

□

Theorem 1. (Forking-Sequences Gradient Variance Reduction)
Consider the forking-sequences gradient estimator $\bar{\nabla} L_T = \frac{1}{T}\sum_t \nabla L_t$. If the gradient samples are M-dependent, then the estimator converges in probability to the true gradient μ at a rate $\mathcal{O}\left(\frac{1}{T}\right)$.

Proof. Using a change of variable and the Markov inequality

$$\mathbb{P}\left(\sqrt{(\bar{\nabla} L_T - \mu)^T \Sigma^{-1} (\bar{\nabla} L_T - \mu)} > \epsilon\right) = \mathbb{P}\left(\sqrt{v} > \epsilon\right) = \mathbb{P}\left(v > \epsilon^2\right) \leq \frac{\mathbb{E}[v]}{\epsilon^2}$$

To bound the numerator, using cyclic property of the trace and the Lemma we know that

$$\begin{aligned}
\mathbb{E}\left[v\right] &= \mathbb{E}\left[(\bar{\nabla} L_T - \mu)^T \Sigma^{-1} (\bar{\nabla} L_T - \mu)\right] = \mathbb{E}\left[\mathrm{tr}\left(\Sigma^{-1}(\bar{\nabla} L_T - \mu)(\bar{\nabla} L_T - \mu)^T\right)\right] \\
&= \mathbb{E}\left[\mathrm{tr}\left(\Sigma^{-1}\mathrm{Cov}(\bar{\nabla} L_T)\right)\right] = \mathbb{E}\left[\mathrm{tr}\left(\Sigma^{-1}\left(\frac{1}{T}\sum_{b=-K}^{K}\left(1 - \frac{|b|}{T}\right)\gamma(b)\right)\right)\right] \\
&\leq \frac{1}{T}\mathbb{E}\left[\mathrm{tr}\left(\Sigma^{-1}\Sigma\right)\right] = \frac{P}{T}
\end{aligned}$$

Combining the two bounds, we obtain that

$$\mathbb{P}\left(\sqrt{(\bar{\nabla} L_T - \mu)^T \Sigma^{-1} (\bar{\nabla} L_T - \mu)} > \epsilon\right) \leq \frac{P}{T\epsilon^2}$$

□

Theorem 2. (*Forking-Sequences Forecast Variance Reduction*) Consider the ensembled forking-sequences forecasts:

$$\hat{\mathbf{y}}^{(q)}_{\tau,\eta} = \frac{1}{|\mathcal{H}|}\sum_{(t,h)\in\mathcal{H}} \hat{\mathbf{y}}^{(q)}_{t,h} \tag{12}$$

If the available of forecasts $\mathcal{H}$ for a target date τ and horizon η are unbiased and M-dependent [43], then the forecast ensemble converges in probability to the true value, and its variance decreases at rate $\mathcal{O}\left(1/|\mathcal{H}|\right)$.

Proof. The proof is analogous to **Theorem 1**. □

E Forking-Sequences Convergence Speed Ablation Study

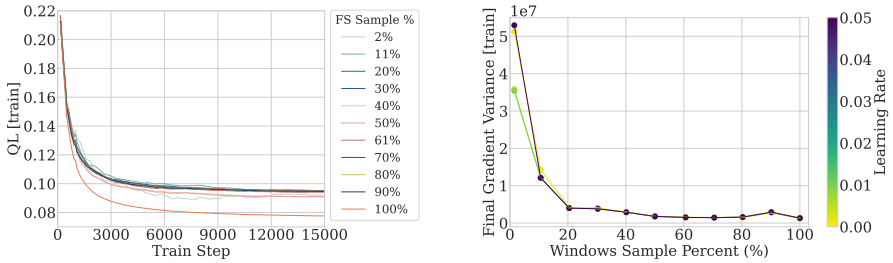
In this ablation study, we examine the impact of the forking-sequences training scheme on the convergence speed of forecasting models. We isolate the effect of the training scheme and ensure convergence to a shared global minimizer, we simplify the experimental setup by replacing the neural network with a linear autoregressive model trained using convex quantile loss (QL; see Equation (2)). An autoregressive model predicts future values as a linear combination of its past values. Specifically, an autoregressive model of order p, denoted AR(p), is defined as:

$$\hat{y}_{t+1} = c + \theta_0 y_t + \theta_1 y_{t-1} + \cdots + \theta_p y_{t-p} \tag{13}$$

We train 11 models on the `M1 Monthly` dataset, with various window sample sizes impacting the choice of the set of FCDs $\mathcal{T}$ over which the training loss and hence the gradient are calculated. Models trained with $\mathcal{T}$ windows are considered to use the default forking-sequences training scheme. For each model we record the computed loss at each training step. To ensure the results are not impacted substantially by the learning rate parameter value, we repeat the experiment for four different learning rates. Model parameters are included in Table 9.

Table 9: Parameters for Linear Autorregresive Model.

Batch size	1
Random seed	1
Maximum Train Steps	15,000
Learning rate decay	0.1 every 1000
Loss quantiles	0.5
Learning rate	[0.001, 0.005, 0.01, 0.05]
Windows Sample Size	[2, 14, 27, 40, 53, 66, 80, 93, 106, 119, 132]

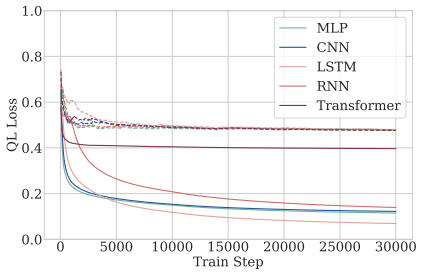


(a) Training Convergence **(b)** Gradient Variance Reduction

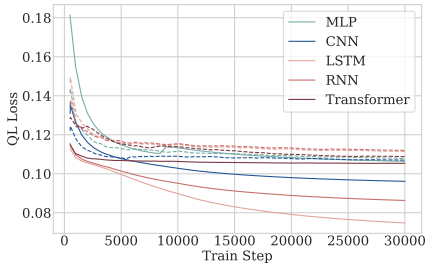
Figure 6: Forking-sequences enables: a) faster convergence in loss on the train set because it b) reduces the variance of the stochastic gradient at a rate of $\mathcal{O}(1/|\mathcal{T}|)$, with $\mathcal{T}$ the number of FCDS. In this example we report the training trajectories for a simple univariate autorregresive model on the M1 dataset. More details in Appendix E.

F Forking-Sequences Convergence Speed Results for Deep Learning Models

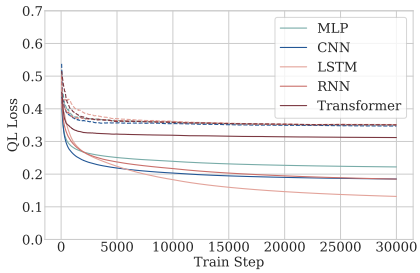
We compare the convergence speed of various deep learning models across dataset experiments using forking-sequences and window-sampling. We find that training using forking-sequences achieves substantially faster convergence in train loss across all models as shown in Fig. 7.



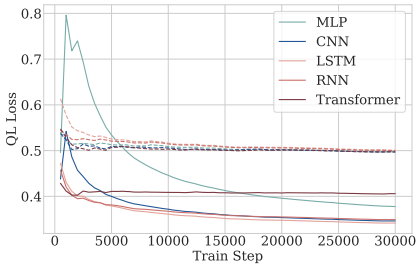
(a) Hourly Frequency Data



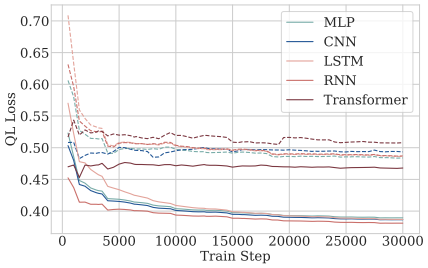
(b) Daily Frequency Data



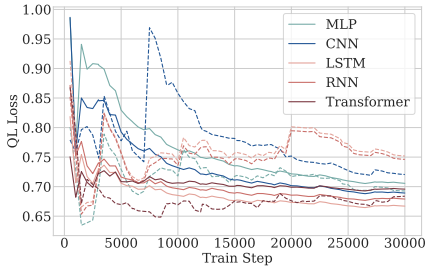
(c) Weekly Frequency Data



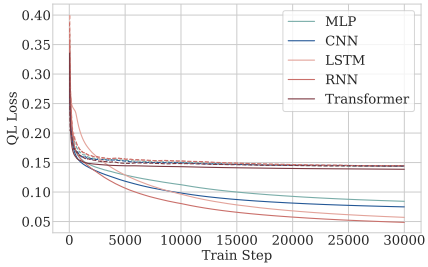
(d) Monthly Frequency Data



(e) Quarterly Frequency Data



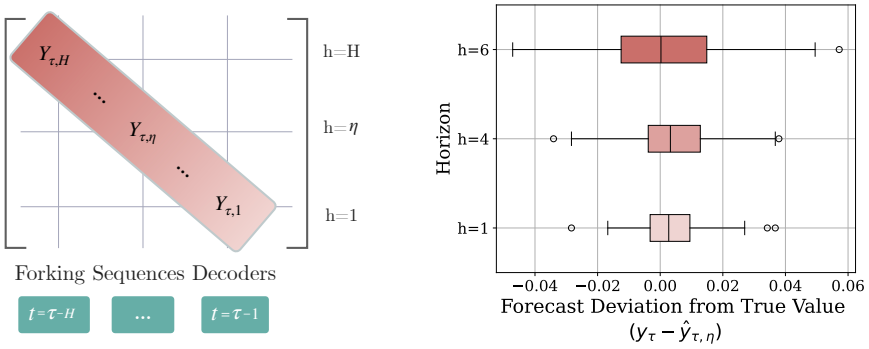
(f) Yearly Frequency Data



(g) Other Frequency Data

Figure 7: Convergence of quantile loss computed on the train set for Deep Learning models using either *forking-sequences* (solid) or *window-sampling* (dashed) techniques. Figures a, b, c, d, e, f show quantile loss versus train step for models trained across frequency-specific datasets.

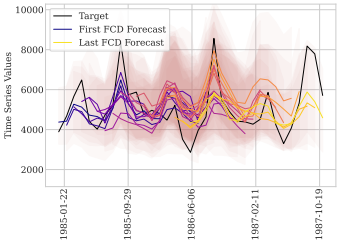
G Ensembling Ablation Studies



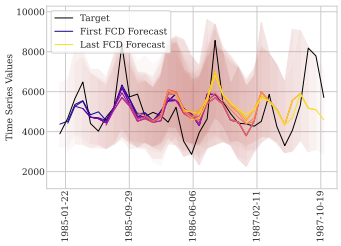
(a) Forking-Sequences Ensemble **(b)** Forecast Variance Reduction

Figure 8: a) We adapt the forking-sequences inference into an ensembling technique by averaging forecasts generated prior to the prediction time. b) As shown in this sub-figure, the forking sequences ensembling approach reduces forecast variance, with a linear convergence rate analogous to the weak law of large numbers.

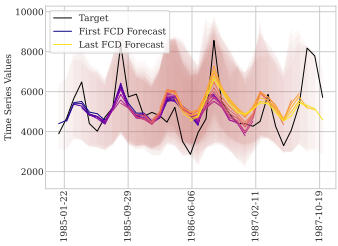
G.1 Forecast Ensembling Techniques



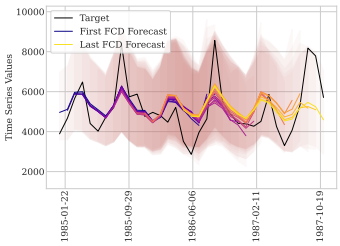
(a) No Ensembling



(b) Moving Median Ensembling



(c) Moving Average Ensembling

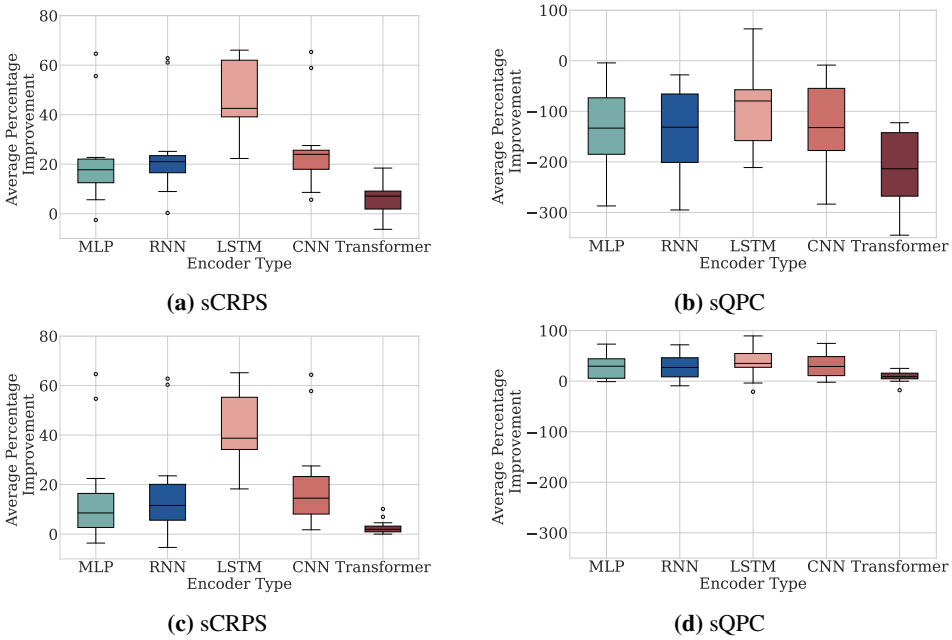


(d) Cumulative Average Ensembling

Figure 9: Forecast outputs for a) no ensembling technique and various ensembling techniques applied during inference, including b) moving median, c) moving average, and d) cumulative average.

We propose the use of ensembling techniques for model inference with the forking-sequences scheme which aggregate predictions for each target date across FCDs to reduce forecast variance. Fig. 9a shows model forecast examples for the M1 monthly dataset 'T10' series without ensembling applied. We demonstrate the output of various ensembling techniques, including rolling median, rolling average, and cumulative average in Figs. 9b, 9c, 9d. Leveraging forecast ensembling during inference results in predictions that maintain seasonality while also reducing forecast variance, as shown in Fig. 9a.

G.2 Forecast Ensembling Impact on Model Predictions



(a) sCRPS **(b)** sQPC

(c) sCRPS **(d)** sQPC

Figure 10: Percentage improvement in sCRPS and sQPC metrics for models with the *forking-sequences* scheme **without ensembling during inference** compared with the *window-sampling* scheme, averaged across datasets **(a, b)**. Percentage improvement in sCRPS and sQPC metrics for models with the *forking-sequences* scheme **with ensembling during inference** compared with the *window-sampling* scheme, averaged across datasets **(c, d)**. Results greater than zero indicate lower metric values using forking-sequences. Forking-sequences with ensembling during inference demonstrates significant improvements in sCRPS and sQPC values over windows-sampling across all encoders. Comparing top and bottom figures highlight a tradeoff in reduced forecast accuracy (sCRPS) for significant reductions in forecast variance (sQPC) when using forecast ensembling during inference.

H Additional and Point Forecasting Results

Table 10: Empirical evaluation of probabilistic forecasts. Mean Absolute Error (MAE) averaged over 5 runs. Lower measurements are preferred. The methods without standard deviation have deterministic solutions. For the `MQForecaster` architecture we vary the type of encoder, and the training scheme between *forking-sequences* (FS) and *window-sampling* (WS).

		`MLP`		`RNN`		`LSTM`		`CNN`		Transf.		StatsForecast	
	Freq	FS	WS	FS	WS	FS	WS	FS	WS	FS	WS	`ETS`	`ARIMA`
M1	M	1787.6 (32.7)	2326.9 (12.1)	1766 (16)	2318.4 (7.2)	1793.8 (42.1)	2757.8 (365.7)	1769.4 (6.1)	2349.8 (11.5)	2184.2 (18.6)	2326 (19)	1898.7 (-)	1987.1 (-)
	Q	2440.4 (53.1)	2645.4 (18.7)	2350.3 (36)	2640.2 (66.7)	2367.7 (43.8)	2956.5 (312.1)	2468.5 (21)	2564.5 (110.8)	2517.4 (34.4)	2657.4 (48.4)	2465.6 (-)	2745.7 (-)
	Y	92089.2 (1876.2)	79943.9 (199)	98461.4 (2667.1)	82307.7 (2386.3)	99082.7 (3012.4)	214505.7 (55460.5)	102777.7 (1866.4)	97485 (8878.6)	101558.4 (5692.6)	78794.4 (1272.6)	100188.5 (-)	96408.3 (-)
M3	O	224.2 (0.9)	222.1 (0.6)	220.4 (1.1)	228.5 (4.1)	237.2 (6.2)	582.1 (47.7)	229.8 (2.6)	238.9 (15.9)	219.9 (2.7)	222.4 (0.9)	203 (-)	206.6 (-)
	M	630.5 (14.3)	744.7 (1.5)	608.3 (9.1)	750.9 (2.7)	609.1 (7.4)	894.6 (46.3)	615.1 (1.1)	751.4 (4.5)	734 (10.2)	748.1 (3.6)	686.9 (-)	691.3 (-)
	Q	552.1 (1)	597.5 (3)	537.4 (4.7)	601.9 (4.4)	538.4 (4.9)	820.8 (84.7)	538 (1.8)	624.7 (14.6)	593.7 (5.4)	599.6 (3.7)	536.1 (-)	535.3 (-)
	Y	1133.8 (10)	1209 (4.9)	1112.3 (4.1)	1206.3 (22.4)	1101.6 (1.9)	2435 (585.5)	1118.6 (2.4)	1186.3 (12.8)	1207.9 (52.8)	1207.3 (5.8)	1050.8 (-)	1066.8 (-)
M4	H	296.5 (13.9)	891.4 (19.6)	307.3 (26.2)	895.4 (22.2)	332 (46.4)	1044.2 (32)	304.9 (8.8)	920 (18.1)	827.7 (10.8)	928.9 (34)	635.6 (-)	290.8 (-)
	D	190.2 (0.2)	189.5 (0.7)	193.6 (0.9)	199.2 (0.9)	191.7 (2.5)	335.6 (61.3)	191 (0.4)	223 (15.7)	189.3 (0.2)	189.5 (0.8)	168.4 (-)	170 (-)
	W	333.6 (5)	380 (2.5)	317.4 (14.9)	388.7 (3.7)	325.7 (14.6)	585.4 (64.5)	299.2 (3.8)	390.2 (6.2)	368.5 (5.5)	384.5 (3.1)	336.6 (-)	325.4 (-)
	M	571.5 (12.2)	624.3 (3.7)	553.5 (1.9)	631.9 (3.2)	554.2 (6)	790.7 (35.9)	555.2 (1.7)	636.4 (6.1)	616.2 (3.6)	626.3 (2)	560.6 (-)	567.6 (-)
	Q	635.8 (0.6)	680.3 (4.3)	623.3 (1.7)	686.6 (8.3)	618.2 (1.2)	906.5 (89.1)	623 (2)	723.8 (16.7)	672.7 (10.7)	683.6 (5.9)	568.4 (-)	597.1 (-)
	Y	927.6 (7.9)	951.8 (2.4)	912.4 (1.2)	949.9 (12.3)	908.2 (0.8)	2061.4 (535.8)	916.4 (1.1)	987.2 (22.5)	968.5 (48.7)	953 (1.6)	849 (-)	889.3 (-)
Tourism	M	2128 (202)	5240.9 (22.4)	1906.3 (81.7)	5269.7 (37.5)	2102.2 (238.1)	6006.2 (844.6)	2035.5 (59.5)	5263.8 (26.5)	4965.1 (119)	5268 (49.3)	2624.5 (-)	3125.8 (-)
	Q	11297.4 (124)	13901.7 (358.5)	11177.9 (55.9)	14062.6 (237.4)	10998.5 (138.2)	17045.2 (1528.1)	10624.8 (103.8)	14669 (590.9)	14239.6 (610.4)	14030.4 (240.4)	11375.6 (-)	12567.5 (-)
	Y	87904.6 (1322.7)	84988.7 (395.3)	86582.1 (536.5)	85582.2 (39)	86584.8 (509.4)	121655.5 (15840.7)	86507.4 (388.5)	89358.6 (1975.7)	84656.7 (1049.2)	85026.1 (238.3)	74574 (-)	86727.3 (-)

To complement the probabilistic results in Section 3. We also evaluate median forecasts denoted by $\hat{\mathbf{y}}_{[b][t][h]}$ through the *mean absolute error* (MAE) [26] as described by

$$\mathrm{MAE}\left(\mathbf{y}_{[b][t][h]},\ \hat{\mathbf{y}}_{[b][t][h]}\right) = \frac{1}{B \times T \times H} \sum_{b,t,h} |y_{b,t,h} - \hat{y}_{b,t,h}|. \tag{14}$$

As discussed in Section 3, forking-sequences improve training convergence. In this section, we compare different encoders trained with window-sampling and forking-sequences: (1) `MLP`, (2) `RNN`, (3) `LSTM`, (4) `CNN`, (5) `Transformer`, and statistical baselines (6) `ETS`, (7) `ARIMA`. Statistical methods are implemented using the StatsForecast library [17]. Forking-sequences consistently improves MAE accuracy across M1, M3, M4and Tourismdatasets. These changes can be attributed to improved training convergence, induced by better gradient flow with the forking-sequences training scheme. The results are generally consistent with sCRPS outcomes in Table 1.