Winner-takes-all for Multivariate Probabilistic Time Series Forecasting

Adrien Cortés [*] **Rémi Rehm** **Victor Letzelter** [* 1 2]

Abstract

We introduce `TimeMCL`, a method leveraging the Multiple Choice Learning (MCL) paradigm to forecast multiple plausible time series futures. Our approach employs a neural network with multiple heads and utilizes the Winner-Takes-All (WTA) loss to promote diversity among predictions. MCL has recently gained attention due to its simplicity and ability to address ill-posed and ambiguous tasks. We propose an adaptation of this framework for time-series forecasting, presenting it as an efficient method to predict diverse futures, which we relate to its implicit *quantization* objective. We provide insights into our approach using synthetic data and evaluate it on real-world time series, demonstrating its promising performance at a light computational cost.

1. Introduction

Predicting the weather of the upcoming weekend or the stock prices of next month with perfect accuracy would undoubtedly be useful. Unfortunately, time-series forecasting is a highly ill-posed problem. In many cases, the available input information is insufficient to reduce our uncertainty about the estimation of the underlying stochastic process and the data itself may contain noise. Consequently, the best a forecaster can do is estimate *plausible* future trajectories, ideally along with the probability of each outcome.

Because temporal data are highly structured and typically come with input–output pairs that require no additional manual annotation, autoregressive neural networks have become the de facto standard for forecasting high-dimensional time series from historical data and exogenous covariates (Rangapuram et al., 2018; Salinas et al., 2019; Benidis et al., 2020). To capture predictive uncertainty, practitioners often place an explicit distribution on the model's output and perform maximum likelihood estimation (Salinas et al., 2020; Alexandrov et al., 2020). While such parametric methods can be computationally efficient, they may depend heavily on the choice of distribution family, reducing their flexibility in capturing complex uncertainties (Gneiting & Katzfuss, 2014).

In parallel, the success of general-purpose generative models, especially conditional diffusion models (Ho et al., 2020) such as `TimeGrad` (Rasul et al., 2021a), has led to strong empirical performance in high-dimensional time series forecasting. However, these models tend to be computationally expensive at inference, particularly when multiple *what-if* scenarios, which we will refer to as *hypotheses*, need to be generated in real-time. Moreover, there is often no explicit mechanism to guarantee sufficiently diverse hypotheses within a single model pass.

To address these limitations, we propose `TimeMCL`—a novel approach based on Multiple Choice Learning (MCL) techniques—that produces diverse and plausible predictions via a single forward pass.

Contributions.

- We introduce `TimeMCL`, a new approach for time series forecasting that adapts the Winner-Takes-All (WTA) loss to generate multiple plausible futures.
- We show that `TimeMCL` can be viewed as a *functional quantizer*, and we illustrate its theoretical properties on synthetic data.
- We evaluate our method on real-world benchmarks, demonstrating that `TimeMCL` efficiently produces a diverse set of forecasts with just a few samples in a single forward pass.[1]

2. Related Work

Ambiguity and need for diversity time series forecasting. In recent years, sequence-to-sequence neural networks (Hochreiter & Schmidhuber, 1997; Sutskever et al., 2014;

[*]Equal contribution [1]LTCI, Télécom Paris, Institut Polytechnique de Paris, France [2]Valeo.ai, Paris, France. Correspondence to: Adrien Cortés <ad.cortes43@gmail.com>, Victor Letzelter <letzelter.victor@hotmail.fr>.

Proceedings of the 42nd International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025. Copyright 2025 by the author(s).

[1]Code available at https://github.com/Victorletzelter/timeMCL.

Chung et al., 2014; Torres et al., 2021) have become increasingly effective in time series forecasting, often surpassing classical techniques (Hyndman et al., 2008; Hyndman & Khandakar, 2008). Yet capturing the inherent ambiguity in future outcomes remains a critical challenge, especially in high-dimensional settings (Ashok et al., 2024). Salinas et al. (2020) proposed a probabilistic autoregressive global model capable of fitting and forecasting high-dimensional time series while highlighting the need to quantify uncertainty in predictions. Building on this line of research, Rasul et al. (2021a) introduced a conditional diffusion model that summarizes the past values of the time series into a hidden state, then performs a diffusion process—conditioned on this state—to generate forecasts. Rasul et al. (2021b) retained the conditional architecture but replaced the diffusion mechanism with a normalizing-flow generator. While these methods are capable of modeling uncertainty, computational efficiency remains a crucial factor, particularly in real-time scenarios (Chen & Boccelli, 2018) where multiple plausible futures must be generated. To address these challenges, we introduce a new family of general-purpose autoregressive time series forecasters based on the Winner-Takes-All (WTA) principle, leveraging its quantization properties to produce diverse and realistic forecasts in a single forward.

Optimal vector quantization & Multiple choice learning for conditional distribution estimation. Quantization is concerned with finding the best finitely supported approximation of a probability measure (Bennett, 1948; Du et al., 1999; Pagès, 2015; Chevallier, 2018). In the context of time series forecasting, this translates to quantizing the *conditional* probability distribution of the target time series. Multiple Choice Learning (MCL) with a Winner-Takes-All (WTA) loss (Guzman-Rivera et al., 2012; Lee et al., 2016) provides a practical framework for such conditional quantization through multi-head networks, which act as a fixed set of codevectors (also called *hypotheses*) (Rupprecht et al., 2017; Letzelter et al., 2024; Perera et al., 2024). While MCL has thus far been explored in various applications, notably computer vision tasks (Lee et al., 2017; Rupprecht et al., 2017; Tian et al., 2019), we adapt it here to predict a quantized representation of the conditional probability distribution of future time series values, using a training scheme specifically tailored for this setting.

3. Problem setup and notations

Let $(x_{t:T}) \in \mathcal{X}^{T-t+1}$ represent a multivariate time series on $\mathcal{X} = \mathbb{R}^D$ over time indexes $[\![t, T]\!]$, where $t \in [\![1, T]\!]$. We aim to learn the conditional law

$$p(x_{t_0:T} \mid x_{1:t_0-1}, c_{1:T}), \tag{1}$$

of future values $x_{t_0:T}$ (over the interval $[\![t_0, T]\!]$) given past observations $x_{1:t_0-1}$ and covariates $c_{1:T}$, the latter being omitted in the following for conciseness. Our focus is on scenarios where the conditional distribution may exhibit multiple modes (*multi-modality*), motivating a richer representation than a single-mean regressor.

To address this issue, the goal of probabilistic time series forecasting is to capture *conditional* distributions over future time series given past values with model p_θ, with parameters θ, whose likelihood can be expressed as

$$p_\theta(x_{t_0:T} \mid x_{1:t_0-1}) = \prod_{t=t_0}^{T} p_\theta(x_t \mid x_{1:t-1}) \,. \tag{2}$$

Once trained, ancestral sampling methods can be used to infer sequence-level predictions. Let us illustrate this scheme using for instance hidden-variable based recurrent neural networks (RNNs) from Graves (2013).

When considering hidden-variables-based models, the basic building block of sequence-to-sequence architectures (Sutskever et al., 2014), one often implicitly parametrize the model with, up to a (log) normalization constant,

$$\log p_\theta(x_t \mid x_{t_0:t-1}, x_{1:t_0-1}) = -\ell(f_\theta(h_{t-1}), x_t) \,, \tag{3}$$

where $\ell(\cdot,\cdot)$ be an appropriate loss, *e.g.,* the mean squared error $\ell(\alpha, \beta) \triangleq \|\alpha - \beta\|^2$ and all the context and previous states is encapsulated into a hidden state $h_{t-1} \in \mathcal{H}$. Assuming the vanilla form for recurrent networks, the hidden state propagation is represented by a model s_θ with $h_{t-1} = s_\theta(x_{t-1}, h_{t-2})$, and f_θ is the final projection.

Once trained, the predictions can then be performed by first encoding the past sequence $x_{1:t_0-1}$ with a hidden state by applying recursively s_θ: $h_{t_0} = s_\theta(x_{t_0-1}, \dots, s_\theta(x_2, s_\theta(x_1, h_0)\dots))$, where h_0 is an arbitrary initial hidden state. Then the recurrent model can be *unrolled*, *i.e.,* turned into autoregressive mode by decoding the predicted sequence by applying recursively s_θ, this time over its own predicted outputs, leading the prediction $\hat{x}_{t_0:T} \sim p_\theta(x_{t_0:T} \mid x_{t_0:T-1}, h_{t_0})$. In the following, we will denote $\mathscr{F}_\theta : x_{1:t_0-1} \mapsto \hat{x}_{t_0:T} = \mathscr{F}_\theta(x_{1:t_0-1}) \in \mathcal{X}^{T-t_0+1}$ the unrolled network.

4. `TimeMCL` method

`TimeMCL` leverages the *Winner-Takes-All* principle (Guzman-Rivera et al., 2012; Lee et al., 2016), which was originally introduced to address ambiguous tasks. WTA naturally extends to scenarios in which future time-series trajectories exhibit *multimodality* (*e.g.,* seasonality, regime switches, and abrupt events). We build on the *Multiple Choice Learning* framework, which produces K distinct "hypotheses" through multiple heads. Not only does our estimator allow us to produce plausible hypotheses; `TimeMCL` provably *quantizes* the target distribution of futures, and is therefore expected to infer the K most representative predictions of the target distribution.

4.1. Training scheme

A key insight behind WTA is that learning K separate hypotheses $\hat{x}^1, \ldots, \hat{x}^K \in \mathcal{X}^{T-t_0+1}$ with an objective that effectively induces a *tesselation* (Du et al., 1999) of the target space into K cells (one for each hypothesis) aims to capture the best possible information of the target distribution with K points.

TimeMCL works as an alternative to the vanilla maximum likelihood estimation (MLE) of (2). Let $p_\theta^1, \ldots, p_\theta^K$ be K models with parameters $(\theta_1, \ldots, \theta_K)$, for which one can associate *heads* $f_\theta^1, \ldots, f_\theta^K : \mathcal{H} \to \mathcal{X}$ using the hidden-state representation as in Section 3. In our implementation, the K models have shared s_θ for hidden-state propagation, and differ only by their final heads f_θ^k, and one may define the complete models with $f_\theta^k \circ s_\theta$.

The Winner-Takes-All consists of the following training scheme for each data point $(x_{1:t_0-1}, x_{t_0:T})$:

1. We compute the negative-log-likelihood of each model

$$\mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T}) = -\sum_{t=t_0}^{T} \log p_\theta^k(x_t \mid x_{1:t-1}) \ ,$$

   where $\log p_\theta^k(x_t \mid x_{1:t-1}) = -\ell(f_\theta^k(h_{t-1}), x_t)$, for each head $k \in [\![1, K]\!]$.

2. We pick the "winner" $k^\star = \arg\min_k \mathcal{L}_\theta^k$, and we back-propagate *only* through that winning head ($k^\star$).

This two-step optimization allows to optimize the loss in an alternating fashion, despite the non-differentiability of the min operator. Note that the latter can be computed batch-wise on the historical data, computing the Winner head of each batch index, with a loss that writes as:

$$\mathcal{L}^{\mathrm{WTA}}(\theta_1, \ldots, \theta_K) \triangleq \mathbb{E}_{x_{1:T}}\big[\min_{k=1,\ldots,K} \mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T})\big] \ , \tag{4}$$

where the expectation is taken over $(x_{1:t_0-1}, x_{t_0:T}) \sim p(x_{1:T})$. While the WTA Loss trains several models with the aim of producing several trajectories, we also use *score* heads as in Letzelter et al. (2023) $\gamma_\theta^1, \ldots, \gamma_\theta^K : \mathcal{H} \to [0, 1]$ to learn to predict the probability of head being the winner and avoid overconfident heads. The latter are trained with

$$\mathcal{L}^{\mathrm{s}} \triangleq \mathbb{E}_{x_{1:T}}\left[\sum_{\substack{k=1,\ldots,K \\ t=t_0,\ldots,T}} \mathrm{BCE}\left(\mathbb{1}\left[k = k^\star\right], \gamma_\theta^k(h_{t-1})\right)\right] \ , \tag{5}$$

where the binary cross entropy $\mathrm{BCE}(p, q) \triangleq -p \log(q) - (1-p)\log(1-q)$, aligns the predicted and target winner assignation probabilities. The final training objective is a compound loss $\mathcal{L} = \mathcal{L}^{\mathrm{WTA}} + \beta \mathcal{L}^{\mathrm{s}}$, where $\beta > 0$ is the confidence loss weight. See Figure 1 for an illustration of the components of TimeMCL.

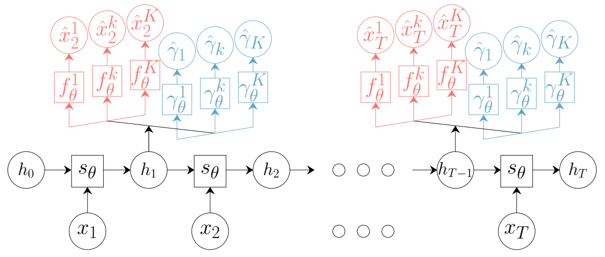


Figure 1. **Components of TimeMCL.** Prediction heads are in Lightcoral color, and Score heads are in Blue. Rectangles contain functions, and circles contain features.

4.2. Inference and Sampling

Once trained, TimeMCL provides K plausible predictions $\hat{x}^1 \sim p_\theta^1, \ldots, \hat{x}^K \sim p_\theta^K$, by ancestral sampling on each of the models. When using recurrent neural networks, the unrolling procedure described in Section 3 can be applied by first encoding the input sequence s_θ to obtain a hidden state h_{t_0}, and unrolling the autoregressive model. As in Section 3, we encapsulate these operation with unrolled networks $\mathscr{F}_\theta^1, \ldots, \mathscr{F}_\theta^K$. The scores are computed in the same way, using the score heads γ_θ^k instead of the prediction heads f_θ^k, and we denote Γ_θ^k their unrolled networks. To get a single score associated with each predicted trajectory, we averaged the predicted scores over the sequence.

In cases where the ambiguity is reduced, such that for short-horizon forecasts when only one prediction is required, one might pick the best head according to the predicted score or sample from them in proportion to some confidence measure. In a longer autoregressive forecast, we can consider the K outputs at each time step —thus producing a set of possible futures from the single forward pass.

4.3. Taking advantage of Winner-takes-all variants

While the WTA Loss has been proven effective for handling ambiguous tasks (Lee et al., 2016; Seo et al., 2020; Garcia et al., 2021), some heads may theoretically be under-trained (Rupprecht et al., 2017). This may occur when a single mode (or a few modes) dominates the target distribution, or due to suboptimal initialization, similar to what can happen in Lloyd's algorithm for K-Means clustering (Lloyd, 1982; Arthur & Vassilvitskii, 2007). In this case, the scoring loss $\mathcal{L}^{\mathrm{s}}$ ensures setting a low probability to those concerning hypotheses so that the latter can be ignored at inference.

It is possible to mitigate this issue, and therefore improve the performance of the estimator by using *relaxation* techniques of the $\min$ operator. In this case, the best head loss in (4) can be substituted with weighted loss from the different heads:

$$\tilde{\mathcal{L}}^{\mathrm{WTA}}(\theta_1, \ldots, \theta_k) \triangleq \mathbb{E}_{x_{1:T}} \left[\sum_{k=1}^{K} q_k \mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T}) \right] , \tag{6}$$

where the coefficients $q_k \geq 0$ sum to one, and allow to assign some weight to non-winner hypotheses.

This idea was originally suggested through Relaxed Winner-takes-all (R-WTA) loss proposed by Rupprecht et al. (2017), which suggested back-propagating not only on the winning head $k^\star$, but also on the non-winner. In this case $q_{k^\star} = 1-\varepsilon$ and $q_k = \frac{\varepsilon}{K-1}$ for $k \neq k^\star$ (See (21) in Appendix C.3). More recently, Perera et al. (2024) proposed an annealed method inspired from Deterministic annealing (Rose et al., 1990) using a *softmin* operator:

$$q_k(T) = \frac{1}{Z(x_{1:t_0-1}, x_{t_0:T}; T)} \exp\Big(-\frac{\mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T})}{T}\Big), \tag{7}$$

with

$$Z(x_{1:t_0-1}, x_{t_0:T}; T) \triangleq \sum_{s=1}^{K} \exp\Big(-\frac{\mathcal{L}_\theta^s(x_{1:t_0-1}, x_{t_0:T})}{T}\Big) ,$$

where the temperature T is annealed during training, *e.g.,* considering at training step t, $T(t) = T(0)\rho^t$ with $\rho < 1$. At higher temperatures, the target assignment is effectively softened, making the early stages of training easier.

We implemented these two variants, within our `TimeMCL` model in the `Gluonts` framework (Alexandrov et al., 2020). Based on our experience, these methods are meaningful variations of WTA that were worth exploring, as they demonstrated improvements over vanilla WTA in certain configurations.

5. Theoretical analysis and interpretation

In this section, based on the notion of functional quantization, we provide insights into the interpretation of our approach. In particular, we show that, under squared error, the K heads form a *Voronoi* tessellation of future trajectories and act as a *conditional codebook*. This viewpoint explains how WTA theoretically captures the best possible way the conditional law over stochastic processes, given a sampling budget of K predictions. Our claims are then illustrated through a synthetic data example, specifically focusing on certain Gaussian Processes.

5.1. **`TimeMCL`** is a stationary conditional functional quantizer

For simplicity and without loss of generality, let us temporarily assume that $\beta = 0$, *i.e.,* only the WTA Loss $\mathcal{L}^{\mathrm{WTA}}$ is optimized. When predicting the future of a time series given its context $x_{1:t_0-1}$, we are effectively observing, during supervision, a (partial) path realization $x_{t_0:T}$ of an underlying stochastic process. Following standard functional data analysis (Bosq, 2000; Ramsay & Silverman, 2005), we assume that each time-series trajectory $x_{t_0:T}$ belongs to a Banach space $(E, \|\cdot\|)$. For concreteness, one may consider $E = L^2([t_0, T])$ endowed with the usual L^2 norm, though any separable Banach space is admissible for the theoretical arguments to hold. In this space, the distance $d(\cdot, \cdot)$ induced by $\|\cdot\|$ enables us to define the Voronoi tessellation over future paths $x_{t_0:T}$ as described below.

The L^2 quantization problem aims at finding the best approximation of a random vector X, using K points in E. The quality of the approximation using at most K points $\{f_1, \ldots, f_K\}$, is generally measured with the distortion, defined as: $D_2(X) = \mathbb{E}_X[\min_{f\in\{f_1,\ldots,f_K\}} d(X, f)^2]$, which is finite over if X admits a second order moment. Note that our learning setup involves quantization in *conditional* form, *i.e.,* the random variable of interest depends on a context.

We state hereafter our main theoretical result, showing that `TimeMCL` can provably perform functional quantization of the target space of plausible trajectories. It can be seen as an adaptation of Proposition 5.2 in Letzelter et al. (2024) to the case of functional quantization of time series.

Proposition 5.1. *See Proposition A.4 in the Appendix. Under the assumptions that:*

1. *The batch size is big enough so that the difference between the $\mathcal{L}^{\mathrm{WTA}}$ risk and its empirical version can be neglected (Assumption A.1).*
2. *The neural network we are considering is expressive enough so that minimizing the risk is equivalent to minimizing the input-dependent risk for each context $x_{1:t_0-1}$ (Assumption A.2).*
3. *The training has converged and $\mathcal{L}^{\mathrm{WTA}}$ has reached a local minima (Assumption A.3).*

Then, `TimeMCL` *is a conditional stationary quantizer for each sampled window $(x_{1:t_0-1}, x_{t_0:T})$, that is, for each $k \in [\![1, K]\!]$:*

$$\mathscr{F}_\theta^k(x_{1:t_0-1}) = \mathbb{E}[x_{t_0:T} \mid x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})] , \tag{8}$$

where

$$\mathcal{X}_k(x_{1:t_0-1}) = \Big\{ x_{t_0:T} \mid \mathcal{L}_\theta^k < \mathcal{L}_\theta^r , \forall r \neq k \Big\} .$$

We denoted by abuse of notations $\mathcal{L}_\theta^k = \mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T})$ for simplicity and the same for $\mathcal{L}_\theta^r$. This makes `TimeMCL` *akin to a conditional and gradient-based version of K-Means over the set of possible future trajectories.*

Sketch of proof. See Proposition A.4 for the full proof. The demonstration of this result is made by first leveraging Assumption A.1 to re-write the WTA Loss in the form

$$\mathbb{E}_{x_{1:t_0-1}}\left[\sum_{k=1}^{K}\int_{\mathcal{X}_k(x_{1:t_0-1})} \mathcal{L}_\theta^k \, \mathrm{d}p(x_{t_0:T} \mid x_{1:t_0-1})\right] , \quad (9)$$

where each $x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})$ picks the head k it is closest to. Under the expressivity assumption (Assumption A.2), (9) comes down to optimizing a functional (14) of the hypotheses position for each fixed context $x_{1:t_0-1}$. We assume that, during training, our predictor generates trajectories solely from the context (*i.e.,* independently of the observed values), effectively as though $\mathscr{F}_\theta^1, \ldots, \mathscr{F}_\theta^K$ were used directly for training.

From here, we leverage the fact that `TimeMCL` is a two-step training procedure, and then the alternating optimization argument from Rupprecht et al. (2017) (Theorem 1) applies to obtain the optimal centroids. This is performed using L^2 square loss for ℓ, from the vanishing gradient condition on the optimized functional (Assumption A.3). We also say that in this case, Voronoi tesselation on the trajectory space induced by the hypothesis is centroidal (Du et al., 1999). □

Now if we assume $\beta > 0$, we can show the following proposition.

Proposition 5.2. *Under similar assumptions as in Proposition 5.1, one can show that a necessary optimality condition for the score heads is that*

$$\Gamma_\theta^k\left(x_{1:t_0-1}\right) = \mathbb{P}\left(x_{t_0:T} \in \mathcal{X}_k\left(x_{1:t_0-1}\right) \mid x_{1:t_0-1}\right). \quad (10)$$

Proof. Full proof in Appendix A.6. □

`TimeMCL` can thus be viewed as a *conditional vector quantization* scheme (Gersho & Gray, 1992), where each head k is a *code vector* (in functional form).

By conditioning on the past data $x_{1:t_0-1}$, `TimeMCL` effectively learns a family of partitions in the time-series trajectory space. If the number of hypotheses is large, increasing K under the above assumptions typically reduces reconstruction error at a rate akin to $K^{-2/d}$, in line with classical quantization theory (Zador, 1982).

5.2. Smoothness of `TimeMCL` predictions for (most) time series

As we have shown, under certain assumptions, if the model reaches a stationary point,

$$\mathscr{F}_\theta^k\left(x_{1:t_0-1}\right) = \mathbb{E}\left[x_{t_0:T} \mid x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})\right] ,$$

and we can interpret the prediction as a mean of different hypothetical trajectories. Since most stochastic time series contain centered noise—caused by various factors such as measurement errors or random events—the averaging process tends to eliminate this noise, resulting in a smooth appearance. We observe this phenomenon consistently in the real examples we visualized. We also noticed that the appearance of smoothness increases as training progresses. This property of *mean* predictions reinforces our conviction that the model is providing representative trajectories rather than just random samples.

5.3. Synthetic data example

We evaluate `TimeMCL` on three synthetic processes: Brownian motion, a Brownian bridge, and an AR(5) process. While the Brownian motion serves as a simple example with minimal context dependence, the Brownian bridge introduces a time-conditioned structure, and the AR(5) process tests the model's ability to handle stronger context dependencies. Training is performed on randomly sampled trajectories with appropriate conditioning, and quantization is assessed using theoretical references: Karhunen-Loève-based quantization for Brownian motion and the Brownian bridge, and Lloyd's algorithm for the AR(5) process (Appendix B).

For this toy experiment, we used a three-layer MLP that predicts the entire sequence at once from its context (see Appendix B for details). To keep the implementation as straightforward as possible, we kept the model's parameterization lightweight and omitted the score heads.

As shown in Figure 2, `TimeMCL` consistently produces smooth and near-optimal quantizations across all settings. For Brownian motion, the predicted trajectories closely align with the theoretical optimal quantization, demonstrating the model's ability to learn conditional distributions from minimal context. In the Brownian bridge setting, where the conditioning depends on both time and value, the model successfully captures the structural constraints, leading to coherent and well-quantized trajectories. The AR(5) process presents a more complex challenge due to its stronger temporal dependencies, yet `TimeMCL` effectively utilizes the past observations to produce long-horizon predictions that remain consistent with Lloyd's quantization. The results highlight the model's ability to condition on past observations, effectively adapting to different processes and maintaining predictive stability over extended time horizons.

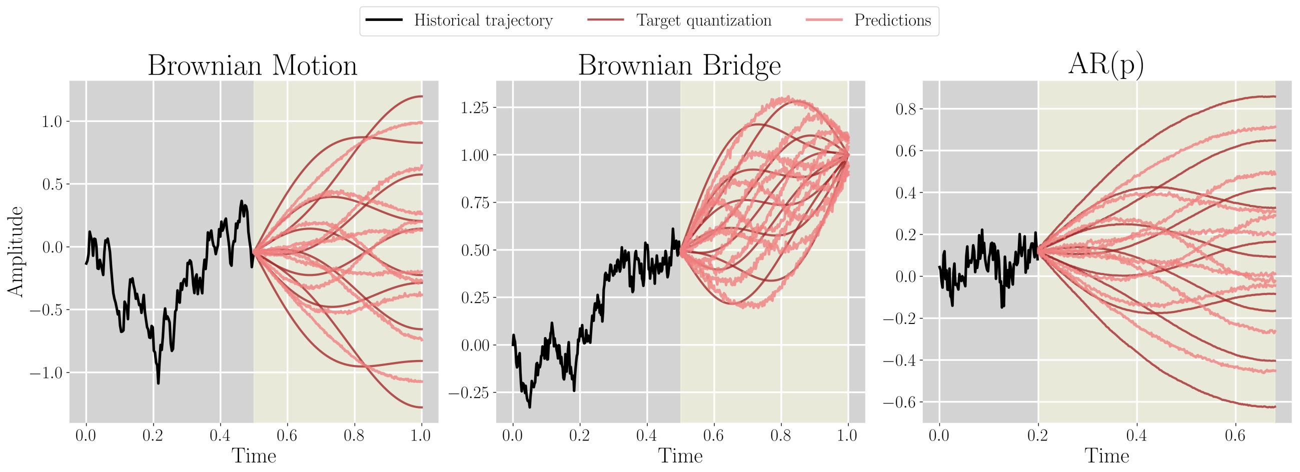


Figure 2. **Conditional Quantization of Stochastic Processes with `TimeMCL`.** The Figure shows the predictions of `TimeMCL` on three synthetic datasets as described in Section 5.3 and Appendix B. Predictions of `TimeMCL` are shown in lightcoral color, and target quantization in each case are shown in brown. We used 10 hypotheses here, a three-layer MLP as backbone with score-heads disabled in those toy experiments. Brownian Motion, Brownian Bridge, and AR(p) are increasingly complex in terms of conditioning dependencies in the context window (See Section 5.3). We see that in those three cases, `TimeMCL` predictions nicely approximate the shape of target quantization functions, justifying its interpretation as a *conditional functional quantizer*.

6. Empirical evaluation

In this section, we empirically validate our method, with experiments on real-world time series. Compared to the experiments in Section 5.3, the underlying law of the data-generating process is not known, making the task more realistic. The goal is to compare `TimeMCL` with state-of-the-art probabilistic time series forecasters, emphasizing its balance between quantization, predictive performance, and computational efficiency.

6.1. Experimental setup

Datasets. Our approach is evaluated on six well-established benchmark datasets taken from `Gluonts` library (Alexandrov et al., 2020), preprocessed exactly as in Salinas et al. (2019); Rasul et al. (2021a). Each dataset consists of strictly positive and bounded real-valued time series. The characteristics of these datasets are summarized in Table 5 in the Appendix. SOLAR contains hourly solar power outputs from 137 sites with strong daily seasonality. ELECTRICITY records hourly power consumption for 370 clients, exhibiting daily and weekly periodicities. EXCHANGE tracks daily exchange rates for eight currencies, showing less seasonality and being influenced by macroeconomic factors. TRAFFIC provides hourly occupancy rates from 963 road sensors, capturing rush-hour peaks and weekly patterns. TAXI comprises time series of NYC taxi rides recorded at 1,214 locations. WIKIPEDIA includes daily views of 2,000 Wikipedia pages (Gasthaus et al., 2019). See Lai et al. (2018) and Appendix C.1 for an extensive description.

Metrics. We evaluate our approach with six different metrics, each of them being detailed in Appendix C.2. First, we considered the Distortion, which is computed as

$$D_2 = \frac{1}{N}\sum_{i=1}^{N} \min_{k=1,\ldots,K} d\left(\mathscr{F}_\theta^k(x^i_{1:t_0-1}), x^i_{t_0:T}\right) \ ,$$

consisting of computing the Euclidean distance d between the target series and closest hypothesis, averaged over the test set with N samples. This allows us to compare fairly with other baselines when the sample size K is fixed. To assess computational efficiency during inference, we report both the inference Floating Point Operations (FLOPs) and run-time for each baseline in Table 2. As a means to validate the theoretical claim presented in Section 5.2, we also compute the Total Variation (TV), which quantifies the smoothness of the predicted trajectories in Table 3. Finally, for comprehensive comparison, we include the Average Root Mean Square Error (RMSE) and the Continuous Ranked Probability Score (CRPS) (summed over all time series dimensions). These results are reported in Tables 7 and 8 in the Appendix.

Baselines. We considered the following baselines: `ETS` (Hyndman et al., 2008), `Tactis2` (Ashok et al., 2024), `DeepAR` (Salinas et al., 2020), `TempFlow` (Rasul et al., 2021b), and `TimeGrad` (Rasul et al., 2021a). These were compared against `TimeMCL` with two relaxation techniques: Relaxed-WTA (Rupprecht et al., 2017) and aMCL (Perera et al., 2024). Note that both of these Multiple Choice Learning variants use score heads as in Letzelter et al. (2023) (with $\beta = 0.5$).

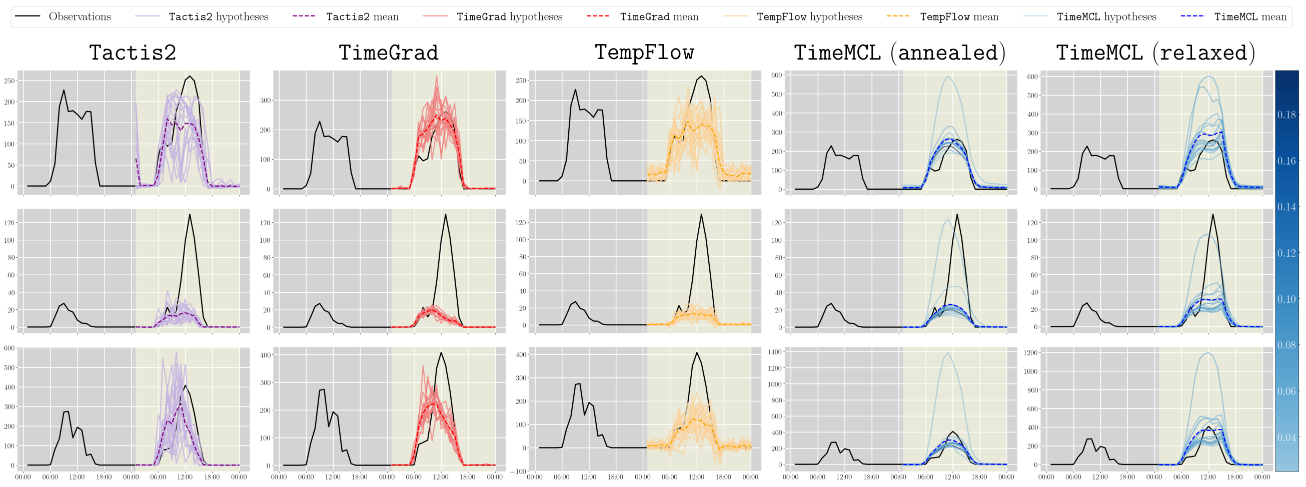


Figure 3. **Qualitative results.** Visualization of the predictions on the SOLAR dataset, comparing `Tactis2` (purple), `TimeGrad` (red), `TempFlow` (orange), and `TimeMCL` trained with two relaxation techniques (blue). Each model predicts sixteen hypotheses, dashed lines represent the predicted mean of the conditional distribution which was computed as the weighted sum of the predicted hypotheses by the scores for `TimeMCL`. Each row represents a different dimension ($D = 137$ here). The light yellow zone highlights the prediction window, and the score intensities are displayed as shaded blue lines, as per the scale on the right. `DeepAR` is not included in this visualization as it does not perform competitively. In this example, `TimeGrad`, `TempFlow` and `Tactis2` generate meaningful hypotheses. However, `TimeMCL`'s predictions are noticeably smoother and more diverse.

Architectures. We compare `DeepAR`, `TempFlow`, and `TimeGrad` with `TimeMCL`, using the same neural network backbone: an RNN with LSTM cells, as in the original implementations (Hochreiter & Schmidhuber, 1997). This ensures fair comparison conditions. In these experiments, each hypothesis head in `TimeMCL` and the projection layer of `DeepAR` consists of a single linear layer. Meanwhile, the noise prediction in `TimeGrad` is implemented with a dilated ConvNet featuring residual connections (Van Den Oord et al., 2016; Kong et al., 2021; Rasul et al., 2021a). Additionally, we include comparisons with methods based on transformer backbones, such as the transformer-based version of `TempFlow`, (named `Trf.TempFlow` (Rasul et al., 2021b)) and `Tactis2` (Drouin et al., 2022; Ashok et al., 2024), which leverages copulas for modeling dependencies. Note that `ETS` does not use a neural network.

Training. Training is conducted using the Adam optimizer with an initial learning rate of 10^{-3} for 200 training epochs. During each epoch, 30 batches of size 200 are sampled from the historical data, considering random windows with a context set equal to the prediction length. We used a validation split of size equal to 10 times the prediction length. Except for `Tactis2`, which uses Z-Score normalization, the data are scaled by computing the mean value dimension by dimension over the context and dividing the target by this mean. This scaling follows the `TimeGrad` experimental setup (Rasul et al., 2021a), ensuring consistency. The model is trained on the scaled data, and the inverse transformation is applied later for prediction.

Evaluation. The evaluation dataset is divided into multiple non-overlapping subsets, each containing sufficient points for both context and prediction lengths, allowing comprehensive assessment across different temporal segments. To compute `TimeMCL` metrics while accounting for predicted hypothesis probabilities, we resample with replacement from the K hypotheses obtained in a single forward pass, weighting them by their predicted scores before computing the metrics.

6.2. Results

Tables 1 and 2 show Distortion performance, FLOPs, and run-time, comparing `TimeMCL` with two tested WTA variants with 16 hypotheses — WTA-Relaxed (Rupprecht et al., 2017) and aMCL (Perera et al., 2024) — against the baselines. Table 3 displays Total Variation comparison, as a means to quantify smoothness.

Distortion and Computation Cost. Table 1 demonstrates that `TimeMCL`, particularly when trained with its Relaxed variant, achieves competitive performance compared to the other models when the number of hypotheses is fixed (with $K = 16$ here). This is especially promising, as `Tactis2` and `TimeGrad`, which are the strongest competitors in terms of distortion, incur significantly higher FLOPs and run-time (see Table 2). A similar trend is observed in Table 6 in the Appendix, which shows results for 8 hypotheses. This behavior is expected since `TimeMCL` explicitly optimizes for distortion. It's worth noting that, among the neural-based methods, `TimeMCL` is the second most com-

Table 1. **Distortion Risk with 16 Hypotheses.** **`TimeMCL(R.)`** and **`TimeMCL(A.)`** correspond to the relaxed and annealed variants. **`ETS`**, **`Trf.TempFlow`** and **`Tactis2`**, columns are in gray because they don't share the same backbone as the other baselines. The test scores are averaged over four training seeds for each model. Best scores are in **bold**, and second-best are underlined.

	`ETS`	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL(R.)`	`TimeMCL(A.)`
ELEC.	14041 ± 877	13519 ± 2789	11616 ± 2160	$\mathbf{9872 \pm 668}$	133107 ± 2870	14836 ± 596	11604 ± 1042	11611 ± 1206
EXCH.	0.051 ± 0.005	0.062 ± 0.015	$\mathbf{0.030 \pm 0.002}$	0.038 ± 0.005	0.061 ± 0.001	0.051 ± 0.007	0.035 ± 0.005	0.044 ± 0.005
SOLAR	641.32 ± 3.57	374.69 ± 20.97	358.01 ± 38.47	360.6 ± 34.79	748.68 ± 18.92	371.14 ± 18.2	$\mathbf{280.03 \pm 15.22}$	305.5 ± 4.05
TRAFFIC	2.64 ± 0.01	1.26 ± 0.02	0.84 ± 0.04	0.78 ± 0.05	2.12 ± 0.08	1.21 ± 0.04	$\mathbf{0.68 \pm 0.01}$	0.72 ± 0.05
TAXI	583.52 ± 0.41	278.22 ± 13.8	243.63 ± 9.1	209.64 ± 3.14	407.4 ± 14.09	268.65 ± 10.69	$\mathbf{187.81 \pm 6.17}$	229.26 ± 22.06
WIKI	715150 ± 4742	515285 ± 16000	$\mathbf{254675 \pm 3870}$	261361 ± 1028	368087 ± 8159	382554 ± 4149	261950 ± 9504	267624 ± 8433

Table 2. **Computational cost at inference of neural-based methods for $K = 16$ hypotheses on EXCHANGE.** FLOPs are computed with a single batch of size 1. Run-times are averaged over 15 random seeds.

	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL`
FLOPS. (↓)	2.04×10^8	1.85×10^8	3.05×10^9	$\mathbf{4.65 \times 10^5}$	9.29×10^7	8.83×10^6
RUN TIME. (↓)	2.47 ± 0.23	8.69 ± 0.36	241.57 ± 2.24	$\mathbf{0.70 \pm 0.04}$	1.39 ± 0.03	1.12 ± 0.04

Table 3. **Total Variation (↓) comparison for $K = 16$ hypotheses.** See Table 11 for the full scores.

	`Tactis2`	`TimeGrad`	`TempFlow`	`TimeMCL(R.)`	`TimeMCL(A.)`
ELEC.	284232 ± 23269	372443 ± 18499	434097 ± 34623	$\mathbf{220375 \pm 33961}$	245908 ± 30505
EXCH.	0.237 ± 0.0355	0.606 ± 0.0208	1.104 ± 0.1582	$\mathbf{0.031 \pm 0.0075}$	0.042 ± 0.0141
SOLAR	4421 ± 828	5383 ± 2173	3653 ± 488	3391 ± 947	$\mathbf{2195 \pm 297}$
TRAFFIC	11.063 ± 0.874	12.991 ± 1.661	18.114 ± 0.512	5.766 ± 0.19	$\mathbf{5.714 \pm 0.283}$
TAXI	5452.77 ± 324.07	4232.16 ± 713.9	4147.2 ± 362.87	709.86 ± 332.07	$\mathbf{701.61 \pm 199.35}$
WIKI	2634547 ± 597323	2458503 ± 151593	12410909 ± 319517	$\mathbf{9532 \pm 10690}$	271611 ± 359320

putationally efficient model, just behind `DeepAR`, while achieving significantly better distortion scores. On that account, `TimeMCL` strikes a promising trade-off between computational cost and performance. For more details, refer to Appendices C.2.4 and C.2.5.

Smoothness. Table 3 displays the Total Variation defined as $\mathrm{TV} = \frac{1}{K}\sum_{k=1}^{K}\sum_{t=t_0}^{T}\left\|\hat{x}_{t+1}^{k} - \hat{x}_{t}^{k}\right\|_2$ which quantifies the average smoothness of the predicted trajectories (lower is more smooth). We see that `TimeMCL`, when trained either with the annealed or relaxed variant, provides significantly smoother trajectories compared to the baselines, further confirming the claim of Section 5.2 as a consequence of Proposition 5.1.

Comparing `TimeMCL` with the baselines on standard metrics. Table 7 and 8 provides performance on CRPS and RMSE, respectively. We see that, except for CRPS and RMSE on EXCHANGE (for which `Tactis` is better), `TimeGrad` outperforms the other baselines. We also observe that `TimeMCL` is competitive despite optimizing a completely different training objective, at a fraction of `TimeGrad`'s and `Tactis`'s computational cost.

Qualitative comparison. We qualitatively compare the predictions of `TimeMCL` with those of the baselines on the SOLAR, TRAFFIC, and ELECTRICITY datasets, as shown in Figure 3 and in Figures 7 and 8 in the Appendix. Figure 3 demonstrates the ability of the hypotheses to predict multiple futures with the aim of capturing different modes of the target data distribution. `Tactis2`, `TimeGrad`, `TempFlow` struggle to produce predictions that deviate significantly from the mean, with most of their predictions being sampled from the same mode of the distribution. Interestingly, `TimeMCL` demonstrates the ability to generate predictions far from the mean with a non-negligible probability, indicating that the model successfully captures different modes and estimates the probability of each mode, including possibly rare events.

6.3. Effect of the number of hypotheses

Table 4 presents a comparison of performance as a function of the number of hypotheses on the test split of the SOLAR dataset. See also Table 10 for an evaluation of the run-time as a function of the number of hypotheses. As expected, the methods generally show improved performance as the number of hypotheses increases, although this also leads to longer run times. However, the performance improvement is not always strictly monotonic with `TimeMCL`. This suggests that our method still has room for refinement, as some hypotheses may remain slightly underutilized. We suspect that this may be partially due to the choice of scaling by the mean, which could be suboptimal for initializing `TimeMCL`. Indeed, as mentioned above, we scale the data by dividing by the mean estimated dimension by dimension, which generally results in a target of constant sign when time series values lie far from the origin, which may be suboptimal when hypotheses are randomly initialized around the origin. To address this, we plan to investigate the impact of different scalers and data pre-processing techniques, such as Z-score normalization or reversible instance normalization (Kim et al., 2021), which could promote better use of hypotheses.

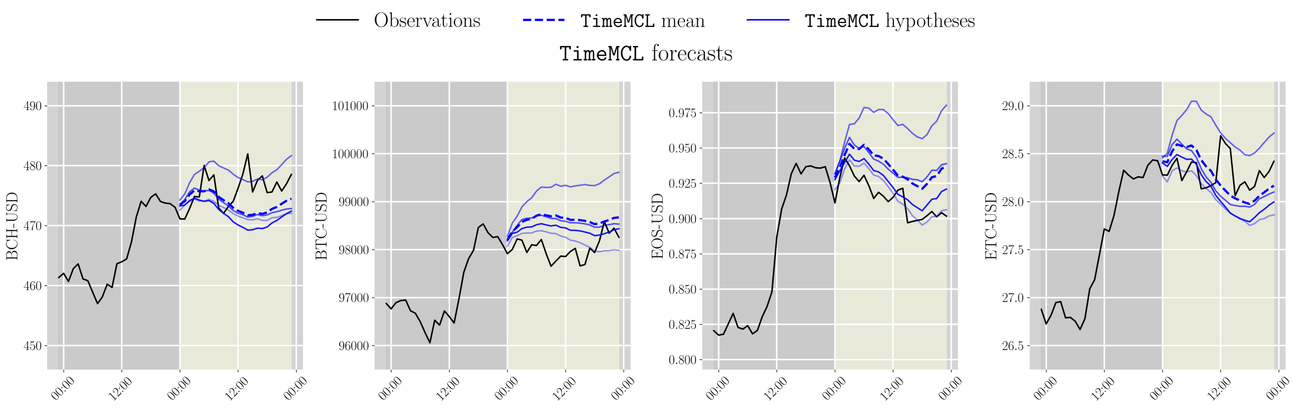


Figure 4. **Cryptocurrency price forecasting for January 4, 2025 with $K = 4$ hypotheses.** Setup in described in Section 6.4. Legend is the same as in Figure 3. We see that `TimeMCL` produces at least one hypothesis that closely matches the realized trajectory. We expect each hypothesis to specialize in a distinct market scenario, for example, a trend reversal or a sudden collapse.

Table 4. **Effect of the number of hypotheses on the Distortion Risk for SOLAR.** Results are averaged over four training seeds. **`TimeMCL(R.)`** and **`TimeMCL(A.)`** corresponds to the relaxed and annealed variants. See Table 9 for the full results.

K	`Tactis2`	`TimeGrad`	`TempFlow`	`TimeMCL(R.)`	`TimeMCL(A.)`
1	433.76 $\pm$ 19.02	**398.96 $\pm$ 32.38**	<u>422.24 $\pm$ 10.64</u>	422.75 $\pm$ 84.28	422.75 $\pm$ 84.28
2	410.23 $\pm$ 12.98	**380.99 $\pm$ 31.06**	404.89 $\pm$ 20.53	<u>384.28 $\pm$ 19.92</u>	418.27 $\pm$ 95.95
3	377.71 $\pm$ 34.55	377.1 $\pm$ 30.87	398.96 $\pm$ 23.97	**356.42 $\pm$ 44.11**	<u>358.48 $\pm$ 20.29</u>
4	376.91 $\pm$ 34.72	375.25 $\pm$ 32.23	386.28 $\pm$ 18.5	**326.62 $\pm$ 20.89**	<u>347.45 $\pm$ 14.59</u>
5	<u>374.76 $\pm$ 32.31</u>	374.96 $\pm$ 32.36	385.54 $\pm$ 19.14	376.25 $\pm$ 43.23	**323.2 $\pm$ 18.3**
8	367.54 $\pm$ 35.4	371.97 $\pm$ 33.31	379.93 $\pm$ 22.38	**305.63 $\pm$ 28.24**	<u>347.82 $\pm$ 28.82</u>
16	358.01 $\pm$ 38.47	360.6 $\pm$ 34.79	371.14 $\pm$ 18.2	**280.03 $\pm$ 15.22**	<u>305.5 $\pm$ 4.05</u>

6.4. Financial time series

`TimeMCL` was evaluated on a corpus of 2 years of hourly cryptocurrency prices (15 correlated tickers collected from `YahooFinance`). Table 5 lists the assets, along with their pairwise correlations and price scales. Due to the wide variance in price magnitudes across assets, we apply Z-score normalization rather than mean scaling. All models are trained with $K = 4$ hypotheses. `TimeMCL` employs the annealed winner-takes-all loss and is compared with `Tactis2`, `TimeGrad`, and `TempFlow`. As reported in Table 12, `TimeMCL` demonstrates strong performance in Distortion, RMSE, and CRPS, while also providing smooth trajectories at a reasonable computation cost (measured in FLOPs). Among the baselines, `Tactis2` and `TempFlow` (with a transformer backbone) achieved competitive results, whereas `TimeGrad`, `DeepAR`, and the RNN version of `TempFlow` were less effective. Figure 4 shows representative trajectories of our approach. At least one `TimeMCL` hypothesis aligns closely with the realized path, illustrating that individual hypotheses can specialize in distinct market regimes (*e.g.,* trend reversals, sudden collapses). Figure 6 contrasts these predictions with the baselines, highlighting both the smoothness of the `TimeMCL` trajectories and their capacity to capture rare events. Implementation details and further analysis can be found in Appendix C.4.

7. Conclusion

We introduced `TimeMCL`, a new model for time series forecasting, designed to predict multiple plausible scenarios that can be viewed as an optimal quantization of the future distribution of the process we aim to predict. This model effectively captures different modes of distribution while providing smooth predictions. Moreover, unlike traditional models that rely primarily on the mean and variance of their predictions for interpretation, `TimeMCL` provides directly interpretable predictions by capturing multiple plausible future scenarios.

`TimeMCL` could be useful when combined with other backbones, as it complements state-of-the-art approaches with its objective function. In this paper, we implemented the model using a recurrent neural network to model temporal dependencies. Exploring the same approach with different architectures and datasets, such as transformer-based backbones when scaling the dataset size, would be a valuable direction for future research.

Limitations. We found that the choice of scaler can be a limitation in `TimeMCL`. An inadequate scaler may bias the model toward certain hypotheses. While relaxation techniques help by softening hard winner assignments, we plan to explore advanced normalization methods to further improve `TimeMCL`'s vanilla setup. Another possible limitation is that `TimeMCL` requires the number of predictions to be predefined beforehand. Exploring dynamic rearrangements of hypotheses when adding new ones without full retraining is left for future work.

Acknowledgements

This work was funded by the French Association for Technological Research (ANRT CIFRE contract 2022-1854) and the LISTEN Laboratory of Télécom Paris. It also benefited from access to the HPC resources of IDRIS (allocation 2024-AD011014345) by GENCI. We are grateful to the reviewers for their insightful comments.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alexandrov, A., Benidis, K., Bohlke-Schneider, M., Flunkert, V., Gasthaus, J., Januschowski, T., Maddix, D. C., Rangapuram, S., Salinas, D., Schulz, J., Stella, L., Türkmen, A. C., and Wang, Y. Gluonts: Probabilistic and neural time series modeling in python. *JMLR*, 2020.

Arcozzi, N., Campanino, M., and Giambartolomei, G. *The Karhunen-Loeve Theorem*. PhD thesis, Master's thesis, University of Bologna, 2015.

Arthur, D. and Vassilvitskii, S. k-means++: the advantages of careful seeding. In *SODA*. Society for Industrial and Applied Mathematics, 2007.

Ashok, A., Marcotte, É., Zantedeschi, V., Chapados, N., and Drouin, A. TACTiS-2: Better, Faster, Simpler Attentional Copulas for Multivariate Time Series. In *ICLR*, 2024.

Asuncion, A., Newman, D., et al. Uci machine learning repository, 2007.

Benidis, K., Rangapuram, S. S., Flunkert, V., Gasthaus, J., Wang, Y., Bohlke-Schneider, M., Salinas, D., Stella, L., Callot, L., and Januschowski, T. Neural forecasting: Introduction and literature overview. *arXiv preprint arXiv:2004.10240*, 2020.

Bennett, W. R. Spectra of quantized signals. *The Bell System Technical Journal*, 1948.

Bosq, D. *Linear Processes in Function Spaces: Theory and Applications*. Springer, 2000.

Chen, J. and Boccelli, D. L. Real-time forecasting and visualization toolkit for multi-seasonal time series. *Environmental Modelling & Software*, 2018.

Chevallier, J. Uniform decomposition of probability measures: quantization, clustering and rate of convergence. *Journal of Applied Probability*, 2018.

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Corlay, S. The nystr\" om method for functional quantization with an application to the fractional brownian motion. *arXiv preprint arXiv:1009.1241*, 2010.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using real nvp. In *ICLR*, 2017.

Drouin, A., Marcotte, É., and Chapados, N. Tactis: Transformer-attentional copulas for time series. In *ICML*, 2022.

Du, Q., Faber, V., and Gunzburger, M. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 1999.

Garcia, N. C., Bargal, S. A., Ablavsky, V., Morerio, P., Murino, V., and Sclaroff, S. Distillation multiple choice learning for multimodal action recognition. In *WACV*, 2021.

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. Probabilistic forecasting with spline quantile function rnns. In *AISTATS*, 2019.

Gersho, A. and Gray, R. M. *Vector Quantization and Signal Compression*. Springer Science & Business Media, Boston, MA, 1992.

Gneiting, T. and Katzfuss, M. Probabilistic forecasting. *Annual Review of Statistics and Its Application*, 2014.

Godahewa, R. W., Bergmeir, C., Webb, G. I., Hyndman, R., and Montero-Manso, P. Monash time series forecasting archive. In *NeurIPS*, 2021.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Guzman-Rivera, A., Batra, D., and Kohli, P. Multiple choice learning: Learning to produce multiple structured outputs. In *NeurIPS*, 2012.

Hendrycks, D. and Gimpel, K. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 1997.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. Neural autoregressive flows. In *ICML*, 2018.

Hyndman, R., Koehler, A. B., Ord, J. K., and Snyder, R. D. *Forecasting with exponential smoothing: the state space approach*. Springer Science & Business Media, 2008.

Hyndman, R. J. and Khandakar, Y. Automatic time series forecasting: the forecast package for r. *Journal of statistical software*, 2008.

Karatzas, I., Shreve, S. E., Karatzas, I., and Shreve, S. E. Brownian motion. *Brownian motion and stochastic calculus*, 1998.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *ICLR*, 2021.

Kong, Z., Ping, W., Huang, J., Zhao, K., and Catanzaro, B. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*, 2021.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long- and short-term temporal patterns with deep neural networks. *SIGIR*, 2018.

Lee, S., Purushwalkam Shiva Prakash, S., Cogswell, M., Ranjan, V., Crandall, D., and Batra, D. Stochastic multiple choice learning for training diverse deep ensembles. In *NeurIPS*, 2016.

Lee, S., Purushwalkam, S., Centeno, M., Zaidan, A., and Batra, D. Confident multiple choice learning. In *ICML*, 2017.

Letzelter, V., Fontaine, M., Chen, M., Pérez, P., Essid, S., and Richard, G. Resilient multiple choice learning: A learned scoring scheme with application to audio scene analysis. In *NeurIPS*, 2023.

Letzelter, V., Perera, D., Rommel, C., Fontaine, M., Essid, S., Richard, G., and Perez, P. Winner-takes-all learners are geometry-aware conditional density estimators. In *ICML*, 2024.

Lloyd, S. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 1982.

Loubes, J.-M. and Pelletier, B. Prediction by quantization of a conditional distribution. *Electronic Journal of Statistics*, 2017.

Matheson, J. E. and Winkler, R. L. Scoring rules for continuous probability distributions. *Management science*, 1976.

Pagès, G. Introduction to vector quantization and its applications for numerics. *ESAIM: Proceedings and Surveys*, 2015.

Pages, G. and Printems, J. Optimal quantization for finance: from random vectors to stochastic processes. In *Handbook of numerical analysis*. Elsevier, 2009.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. Normalizing flows for probabilistic modeling and inference. *JMLR*, 2021.

Perera, D., Letzelter, V., Mariotte, T., Cortés, A., Chen, M., Essid, S., and Richard, G. Annealed multiple choice learning: Overcoming limitations of winner-takes-all with annealing. In *NeurIPS*, 2024.

Ramsay, J. O. and Silverman, B. W. *Functional Data Analysis*. Springer, 2005.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *NeurIPS*, 2018.

Rasul, K., Seward, C., Schuster, I., and Vollgraf, R. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *ICML*, 2021a.

Rasul, K., Sheikh, A.-S., Schuster, I., Bergmann, U. M., and Vollgraf, R. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *ICLR*, 2021b.

Rose, K., Gurewitz, E., and Fox, G. A deterministic annealing approach to clustering. *Pattern Recognition Letters*, 1990.

Rupprecht, C., Laina, I., DiPietro, R., Baust, M., Tombari, F., Navab, N., and Hager, G. D. Learning in an uncertain world: Representing ambiguity through multiple hypotheses. In *ICCV*, 2017.

Salinas, D., Bohlke-Schneider, M., Callot, L., Medico, R., and Gasthaus, J. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *NeurIPS*, 2019.

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2020.

Seo, Y., Lee, K., Clavera Gilaberte, I., Kurutach, T., Shin, J., and Abbeel, P. Trajectory-wise multiple choice learning for dynamics generalization in reinforcement learning. In *NeurIPS*, 2020.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *NeurIPS*, 2014.

Tian, K., Xu, Y., Zhou, S., and Guan, J. Versatile multiple choice learning and its application to vision computing. In *CVPR*, 2019.

Torres, J. F., Hadjout, D., Sebaa, A., Martínez-Álvarez, F., and Troncoso, A. Deep learning for time series forecasting: a survey. *Big Data*, 2021.

Tsay, R. S. Analysis of financial time series. *John Eiley and Sons*, 2005.

Van Den Oord, A., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K., et al. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Zador, P. L. Asymptotic quantization error of continuous signals and the quantization dimension. *IEEE Transactions on Information Theory*, 1982.

Organization of the Appendix

The Appendix is organized as follows. Appendix A contains the proofs of the theoretical results, establishing that `TimeMCL` can be interpreted as a functional quantizer. Appendix B describes the synthetic data used in the toy example. Appendix C details the empirical evaluation on real data, covering the datasets (Appendix C.1), the experimental procedure and evaluation metrics (Appendix C.2), and the baseline models (Appendix C.3). Finally, Appendix C.4 presents additional results for the financial time series.

A. Theoretical results

Following the main paper notations, we assume each time series lives in $\mathcal{X} = \mathbb{R}^D$. We refer to the context and target sequences as $x_{1:t_0-1} \in \mathcal{X}^{1:t_0-1}$ and $x_{t_0:T} \in \mathcal{X}^{t_0:T}$ respectively.

A.1. Proof of Proposition 5.1

Let us consider the following assumptions.

Assumption A.1 (True risk minimization)**.** The batch size is big enough so that the difference between the $\mathcal{L}^{\mathrm{WTA}}$ risk and its empirical version can be neglected.

In the Assumption A.2 that follows, we considered the forecaster (or unrolled) network $\mathscr{F}_\theta$ as a function $\mathcal{X}^{1:t_0-1} \to \mathcal{X}^{t_0:T}$ that is directly optimized during training, *i.e.,* teacher forcing is disabled. This implies that the same forward computation applies in training and inference. In particular, the model predictions on the target window $\mathcal{X}^{T-t_0+1}$ at training time depend only on the past context $x_{1:t_0-1}$. We adopt this simplified setup in the toy experiments presented in Section 5.3 and in Appendix B.

Assumption A.2 (Expressiveness)**.** The family of neural networks considered is expressive enough so that minimizing the expected training risk reduces to minimizing the input-dependent risk for each individual input[2]. Formally, this means

$$\inf_{\theta\in\Theta} \mathbb{E}_{x_{1:T}} \left[\ell\left(\mathscr{F}_\theta(x_{1:t_0-1}), x_{t_0:T}\right)\right] = \int_{\mathcal{X}^{1:t_0-1}} \inf_{z\in\mathcal{X}^{t_0:T}} \mathbb{E}_{x_{t_0:T}\sim p(x_{t_0:T}|x_{1:t_0-1})} \left[\ell\left(z, x_{t_0:T}\right)\right] \, \mathrm{d}p(x_{1:t_0-1}) \; , \tag{11}$$

where Θ is the set of possible neural network parameters, and ℓ may be the mean square error loss $\ell\left(\hat{x}_{t_0:T}, x_{t_0:T}\right) \triangleq \sum_{t=t_0}^{T} \|\hat{x}_t - x_t\|^2$.

Assumption A.3 (Optimality)**.** The training has converged and $\mathcal{L}^{\mathrm{WTA}}$ has reached a local minima.

Proposition A.4. *Under the Assumptions A.1, A.2 and A.3,* `TimeMCL` *is a conditional stationary quantizer for each sampled window* $(x_{1:t_0-1}, x_{t_0:T})$*, that is*

$$\mathscr{F}_\theta^k(x_{1:t_0-1}) = \mathbb{E}[x_{t_0:T} \mid x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})], \tag{12}$$

where

$$\mathcal{X}_k(x_{1:t_0-1}) \; = \; \Big\{x_{t_0:T} \in \mathcal{X}^{t_0:T} \mid \mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T}) < \mathcal{L}_\theta^r(x_{1:t_0-1}, x_{t_0:T}) \; , \forall r \neq k\Big\},$$

where accordingly with Assumption A.2; $\mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T}) = \sum_{t=t_0}^{T} \|\mathscr{F}_\theta^k(x_{1:t_0-1})_t - x_t\|^2$*. This makes* `TimeMCL` *akin to a conditional and gradient-based version of K-Means over the set of possible future trajectories.*

Proof. Assuming we are performing true risk minimization (Assumption A.1), the WTA Loss $\mathcal{L}^{\mathrm{WTA}}$ can be re-written as

$$\mathcal{L}^{\mathrm{WTA}} = \int_{x_{1:t_0-1}\in\mathcal{X}^{1:t_0}} \sum_{k=1}^{K} \int_{x_{t_0:T}\in\mathcal{X}_k(x_{1:t_0-1})} \mathcal{L}_\theta^k(x_{1:t_0-1}, x_{t_0:T}) \, \mathrm{d}p(x_{t_0:T} \mid x_{1:t_0-1}) \mathrm{d}p(x_{1:t_0-1}) \; , \tag{13}$$

due to Chasles relation because the Voronoi tesselation forms a partition of the output space. Under the expressiveness Assumption A.2, (13) comes down to optimizing

$$\mathcal{F}_{x_{1:t_0-1}}(z^1, \ldots, z^K) \triangleq \sum_{k=1}^{K} \int_{\mathcal{X}_k(x_{1:t_0-1})} \mathcal{L}_z^k(x_{1:t_0-1}, x_{t_0:T}) \mathrm{d}p(x_{t_0:T} \mid x_{1:t_0-1}) \; , \tag{14}$$

[2]See also Assumption 1 in Perera et al. (2024) and Section 3 of Loubes & Pelletier (2017).

for each fixed context $x_{1:t_0-1}$, where each $z^k \in \mathcal{X}^{t_0:T}$ and

$$\mathcal{L}_z^k(x_{1:t_0-1}, x_{t_0:T}) \triangleq \sum_{t=t_0}^{T} \ell(z_t^k, x_t) \ .$$

To prove that each model $\mathscr{F}_\theta^k(x_{1:t_0-1})$ converges to the conditional mean of the future trajectories that fall into its corresponding Voronoi cell, we follow the decoupling strategy of Rupprecht et al. (2017) and Letzelter et al. (2024). Define two sets of variables:

- *Generators* $\{g^k(x_{1:t_0-1})\}_{k=1}^K$, which induce a partition of the output space $x_{t_0:T}$ via the Voronoi diagram:

$$\mathcal{X}_k(g) \ \triangleq \ \Big\{x_{t_0:T} \in \mathcal{X}^{t_0:T} \mid \|x_{t_0:T} - g^k(x_{1:t_0-1})\|^2 < \|x_{t_0:T} - g^r(x_{1:t_0-1})\|^2, \ \forall r \neq k\Big\}.$$

- *Centroids* $\{z^k(x_{1:t_0-1})\}_{k=1}^K$, which are the points used to compute the intra-cell L^2 loss.

Decoupling generators vs. centroids through alternating optimization. In `TimeMCL` the update is done in a two-step fashion: first, the winners are computed for each point in the batch, and then the latter are updated. This two-step optimization allows us to bypass the non-differentiability of the $\min$ operator in (4).

Let us *decouple* the two variables and define the following functional to optimize:

$$\mathcal{F}\big(g, z; x_{1:t_0-1}\big) \triangleq \sum_{k=1}^{K} \int_{\mathcal{X}_k(g)} \|x_{t_0:T} - z^k(x_{1:t_0-1})\|^2 \ p(x_{t_0:T} \mid x_{1:t_0-1}) \, \mathrm{d}x_{t_0:T} \ .$$

We now consider minimizing $\mathcal{F}(g, z; x_{1:t_0-1})$ with respect to g and z in an alternating manner:

- **(a) Fixing the partition (g) and optimizing the centroids (z).** In this case, the gradient update of z^k has direction:

$$\frac{\partial \mathcal{F}}{\partial\, z^k(x_{1:t_0-1})}(g, z, x_{1:t_0-1}) = 2\left(z^k(x_{1:t_0-1})\mathrm{Vol}(\mathcal{X}_k(g)) - \int_{\mathcal{X}_k(g)} x_{t_0:T}\, p(x_{t_0:T} \mid x_{1:t_0-1}) \mathrm{d}x_{t_0:T}\right) \ ,$$

  where $\mathrm{Vol}(\mathcal{X}_k(g)) \triangleq \int_{\mathcal{X}_k(g)} p(x_{t_0:T} \mid x_{1:t_0-1}) \mathrm{d}x_{t_0:T}$. Thus each $z^k(x_{1:t_0-1})$ is updated in the direction of the cell conditional mean, similarly to the Lloyd algorithm (Lloyd, 1982).

- **(b) Fixing the centroids (z) and optimizing the partition (g).** Conversely, if z is fixed, one can reduce $\mathcal{F}(g, z; x_{1:t_0-1})$ by ensuring that $\mathcal{X}_k(g)$ is indeed the Voronoi cell generated by $z^k(x_{1:t_0-1})$. Indeed, for any deviation from a strict Voronoi partition, there would exist subsets of $x_{t_0:T}$ incorrectly assigned to some cell, and reassigning them to the nearest $z^k(x_{1:t_0-1})$ would lower the overall loss (See also Rupprecht et al. (2017), Theorem 1).

By alternating between these two steps, any stationary point of $\mathcal{F}(g, z; x_{1:t_0-1})$ must satisfy

$$z^k(x_{1:t_0-1}) \ = \ g^k(x_{1:t_0-1}) \ .$$

Such a configuration is a *centroidal Voronoi tessellation* (see, *e.g.,* Du et al., 1999).

Reverting to the original notation of the WTA network, we identify $g^k(x_{1:t_0-1}) = z^k(x_{1:t_0-1}) = \mathscr{F}_\theta^k(x_{1:t_0-1})$ at optimality, thus each prediction head coincides with the mean of the distribution restricted to its Voronoi cell:

$$\mathscr{F}_\theta^k(x_{1:t_0-1}) \ = \ \mathbb{E}\big[x_{t_0:T} \ \mid \ x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})\big] \ .$$

Consequently, at the global minimum of the WTA objective, we obtain the desired *conditional* centroidal Voronoi tessellation. $\square$

A.2. Proof of Proposition 5.2 (Scoring Heads as Unbiased Estimators)

Once trained, predictions from score heads are performed through the unrolled version of the score heads Γ_θ^k, which can be viewed as a function $\Gamma_\theta^k : \mathcal{X}^{1:t_0-1} \to [0,1]$ when averaging the predicted scores over the sequence. In the following, we assume that the Γ_θ^k is directly optimized during training, similarly to the approach used for the prediction heads in Section A. We prove that when the training of `TimeMCL` with $\beta > 0$ has converged globally, the (unrolled) scoring heads $\Gamma_\theta^k(x_{1:t_0-1})$ match the conditional probability mass of their respective Voronoi region $\mathcal{X}_k(x_{1:t_0-1})$. This statement generalizes the arguments from Letzelter et al. (2024) to our time-series setup.

Assumption A.5 (Global optimality for both centroids *and* scores)**.** In addition to having converged to a global minimum of the centroid objective (13), the `TimeMCL` model also reaches a global minimum for its scoring objective

$$\mathcal{L}^{\mathrm{s}}(\theta) = \int_{\mathcal{X}^{1:t_0-1}} \sum_{k=1}^{K} \int_{\mathcal{X}_k(x_{1:t_0-1})} \mathrm{BCE}\Big(\mathbb{1}[x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})],\, \Gamma_\theta^k(x_{1:t_0-1})\Big)\, p(x_{1:t_0-1}, x_{t_0:T})\, \mathrm{d}x_{t_0:T}\, \mathrm{d}x_{1:t_0-1}\ ,$$

where $\mathrm{BCE}(\cdot,\cdot)$ denotes the binary cross-entropy, and $x_{1:t_0-1} \mapsto (\mathcal{X}_k(x_{1:t_0-1}))_{k=1}^K$ is the optimal Voronoi tessellation corresponding to the converged heads $\{\mathscr{F}_\theta^k\}$.

Proposition A.6 (Unbiased Estimator of Voronoi Cell Mass)**.** *We assume, as in Assumption A.1, that the batch size is big enough so that the difference between the $\mathcal{L}^{\mathrm{s}}$ and its empirical version can be neglected. Under Assumption A.5, and assuming perfect expressiveness of the score heads as in Assumption A.2, for any context $x_{1:t_0-1}$ and index $k \in \{1,\ldots,K\}$, the optimal scoring head satisfies*

$$\Gamma_\theta^k(x_{1:t_0-1}) = \mathbb{P}\left(x_{t_0:T} \in \mathcal{X}_k\left(x_{1:t_0-1}\right) \mid x_{1:t_0-1}\right).$$

Proof. We adapt the derivation of the unbiased property to our time-series setup. Recall that the optimal scoring objective (see Assumption A.5) may be written as

$$\min_\theta \int_{\mathcal{X}^{1:t_0-1}} \sum_{k=1}^{K} \int_{\mathcal{X}_k(x_{1:t_0-1})} \mathrm{BCE}\Big(\mathbb{1}[x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})],\, \Gamma_\theta^k(x_{1:t_0-1})\Big)\, p(x_{t_0:T} \mid x_{1:t_0-1})\, p(x_{1:t_0-1})\, \mathrm{d}x_{t_0:T}\, \mathrm{d}x_{1:t_0-1}. \tag{15}$$

Under the true risk minimization and expressiveness assumptions, and since the input-dependent risk decomposes over each $k \in \{1,\ldots,K\}$, we can focus on a *fixed* $x_{1:t_0-1}$ and a single index k in the sum. For that fixed $x_{1:t_0-1}$ and k, the part of (15) to optimize is:

$$\int_{\mathcal{X}^{t_0:T}} \mathrm{BCE}\Big(\mathbb{1}[x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1})],\, \Gamma_\theta^k(x_{1:t_0-1})\Big)\, p(x_{t_0:T} \mid x_{1:t_0-1})\, \mathrm{d}x_{t_0:T}\ .$$

Writing out the binary cross-entropy explicitly, we get:

$$-\int_{\mathcal{X}_k(x_{1:t_0-1})} \log\big(\Gamma_\theta^k(x_{1:t_0-1})\big)\, p(x_{t_0:T} \mid x_{1:t_0-1})\, \mathrm{d}x_{t_0:T} - \int_{\mathcal{X}^{t_0:T} \setminus \mathcal{X}_k(x_{1:t_0-1})} \log\big(1 - \Gamma_\theta^k(x_{1:t_0-1})\big)\, p(x_{t_0:T} \mid x_{1:t_0-1})\, \mathrm{d}x_{t_0:T}.$$

Let us denote $m_k(x_{1:t_0-1}) = \mathbb{P}\big(x_{t_0:T} \in \mathcal{X}_k(x_{1:t_0-1}) \mid x_{1:t_0-1}\big)$. Then the above expression becomes

$$-\, m_k(x_{1:t_0-1})\, \log\big(\Gamma_\theta^k(x_{1:t_0-1})\big) \;-\; (1 - m_k(x_{1:t_0-1}))\, \log\big(1 - \Gamma_\theta^k(x_{1:t_0-1})\big)\ .$$

We recognize this as the binary cross-entropy between the probabilities $\Gamma_\theta^k(x_{1:t_0-1})$ and $m_k(x_{1:t_0-1})$. The unique global minimum of that scalar binary cross-entropy is attained at

$$\Gamma_\theta^k(x_{1:t_0-1}) \;=\; m_k(x_{1:t_0-1}) \;=\; \mathbb{P}\big(x_{t_0:T} \in \mathcal{X}_k\left(x_{1:t_0-1}\right) \mid x_{1:t_0-1}\big)\ .$$

This equality must hold for each context $x_{1:t_0-1}$ and each index k, completing the proof. □

B. Experimental details on the synthetic datasets

All toy examples were conducted using a neural network with a three-layer fully connected backbone (200 hidden units, ReLU activation) and K prediction heads, each generating a trajectory of a given length. Training uses a Winner-Takes-All loss combined with squared error minimization, optimized with Adam using a learning rate of 10^{-3}, a batch size of 4096, and 500 iterations. For each of the setups, we used the Relaxed version of `TimeMCL` with $\varepsilon = 0.05$.

Brownian Motion. In order to conduct interpretable experiments and generate synthetic data, we employed Brownian motion. A Brownian motion $\{W_t\}_{t\geq 0}$ is a Gaussian process characterized by a mean $\mathbb{E}[W_t] = 0$ and a covariance function $\mathrm{Cov}(W_t, W_s) = \min(t, s)$ for $t, s \geq 0$. This process possesses several properties of interest (Karatzas et al., 1998). In particular, the Markov property, which ensures that the process $\{W_{s+t} - W_s\}_{t\geq 0}$ is itself a Brownian motion starting at 0, independent of the process $\{W_t : 0 \leq t \leq s\}$. This implies that to condition a Brownian motion on its trajectory up to time t, it suffices to condition on its value W_t. The process beyond time t behaves as if it were a new Brownian motion originating at W_t. In conditional quantization, we aim to quantize Brownian motion at a given prediction horizon, conditioned on its value at time t. To compute the target quantization, we followed the methodology detailed in (Pages & Printems, 2009), leveraging the Karhunen-Loève decomposition (Arcozzi et al., 2015) of the Brownian motion on $t \in [0, 1]$:

$$W_t = \sum_{n=1}^{\infty} \sqrt{\lambda_n^W} \xi_n e_n^W(t) \ , \tag{16}$$

where $\xi_n \sim \mathcal{N}(0, 1)$ are independent standard Gaussian random variables. Here, $\lambda_n^W \triangleq \frac{1}{\pi^2 (n-\frac{1}{2})^2}$ are the eigenvalues of the covariance operator of Brownian motion, and $e_n^W(t) \triangleq \sqrt{2}\sin(\pi(n - \frac{1}{2})t)$ are the corresponding eigenfunctions (Corlay, 2010). To quantize Brownian motion, we first truncate its Karhunen-Loève decomposition (16) to m terms. Then, the coefficients ξ_n are quantized into K_n optimal levels $\{\alpha_1^{(K_n)}, \dots, \alpha_{K_n}^{(K_n)}\}$, using the quantile function of the normal distribution. All possible combinations of quantized coefficients are generated via the Cartesian product. In this case, for each multi-index $i = (i_1, \dots, i_m) \in \prod_{n=1}^{m} [\![1, K_n]\!]$, it is possible to define a quantizer

$$x_i^{(K_1,\dots,K_m)}(t) \triangleq \sum_{n=1}^{m} \sqrt{\lambda_n^W} \alpha_{i_n}^{(K_n)} e_n^W(t) \ ,$$

which takes $\prod_{n=1}^{m} K_n$ possible values. Following Pages & Printems (2009), product quantization with exactly K codevectors requires optimizing two parameters: the truncation parameter m and the quantization level allocation $\{K_n\}_{n=1}^{m}$, with $\prod_{n=1}^{m} K_n = K$. Pages & Printems (2009) provides a list of optimal tuples for these parameters. In our numerical example, we select $m = 2$ and $(K_1, K_2) = (5, 2)$ resulting in a quantizer with $5 \times 2 = 10$ codevectors.

`TimeMCL` was trained on 250-step sequences randomly drawn from a Brownian motion on $[0, 1]$, discretized with 500 steps. At each iteration, a new sample was generated, with the model conditioned on the last observed value and tasked with predicting the next 249 steps. The generated trajectories are smooth and closely match the theoretical optimal ones. The model demonstrates consistent long-term predictions and effectively learns conditional quantization.

Brownian Bridge. Brownian motion is unrepresentative of most natural processes, as its future evolution is independent of the conditioning time. In contrast, many real-world processes depend on when they are observed. A Brownian bridge, viewed as a Brownian motion conditioned to reach a fixed value at $T = 1$, better captures such dependencies. A continuous process $\{B_t\}_{t\in[0,T]}$ is a Brownian bridge on the interval $[0, T]$ if and only if it has the same distribution as $\{W_t - \frac{t}{T} W_T\}_{t\in[0,T]}$, where $\{W_t\}_{t\in[0,T]}$ is a standard Brownian motion. In our case, $T = 1$, and the process B is referred to as a standard Brownian bridge. As we need to conditionally quantize the Brownian bridge, we note that if we pick a random time $t \in [0, 1]$, the Brownian bridge conditioned on its value at this point remains a Brownian bridge. To obtain the optimal quantization, the same process as for Brownian motion is used, with adapted eigenvalues and eigenvectors in the Karhunen-Loève decomposition (See Corlay (2010), page 3),

$$e_n^B(t) \triangleq \sqrt{\frac{2}{T}} \sin\left(\pi n \frac{t}{T}\right), \quad \lambda_n^B \triangleq \frac{T^2}{\pi^2 n^2} \ , \quad n \geq 1 \ .$$

In our numerical example, we select the same values for m and (K_1, K_2) as those chosen for the Brownian motion. `TimeMCL`, like for Brownian motion, is trained on random samples from a Brownian bridge starting at $B_0 = 1$ and ending at $B_1 = 1$. With 500 discretization points, it predicts 250 steps, conditioned on the prediction start time and value.

Autoregressive model AR(p). The two first experiments involved Brownian motion, where only the last value matters for prediction, and the Brownian bridge, which depends on the last observed time and value. While these processes assess `TimeMCL`'s quantization ability, they lack strong context dependence. Real-world applications require modeling context to predict and quantize future trajectories effectively. Consider an autoregressive process

$$X_t = \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t \ , \tag{17}$$

where $(\epsilon_t)_t$ is a white noise with variance σ^2. We generate an autoregressive sequence where the first five values follow a normal distribution with scale σ. The process runs for an initial warm-up period of 100 steps. After this, we predict and quantize the next 250 steps using the last 100 observations, i.e., $\{X_{t_0-99}, \dots, X_{t_0}\}$. The five values preceding t_0 form a crucial context—without it, correctly quantizing the future distribution is impossible. `TimeMCL` is expected to exploit this information for optimal quantization. To the best of our knowledge, no simple analytical solution exists for quantizing an autoregressive process. To approximate an optimal quantization, the process is simulated multiple times from t_0, conditioned on its context, to construct a conditional distribution. The Lloyd algorithm, run with a very large number of simulations (10^5 sampled trajectories), produces a set of *quantized* trajectories that serve as references for `TimeMCL`.

`TimeMCL` is trained on 250-step sequences randomly sampled from 500-step trajectories of the AR(5) process, with a new batch generated at each iteration. To ensure consistency with Brownian bridge and motion experiments, time is normalized by dividing all steps by the total number of steps. We took $\phi_1 = 0.4, \phi_2 = \phi_3 = 0.2, \phi_4 = \phi_5 = 0.1$, and $\sigma = 0.06$. The trajectories predicted by `TimeMCL` closely align with the optimal ones found by Lloyd's algorithm, exhibiting smooth behavior. Various values and orders of the autoregressive process, both stationary and non-stationary, were tested, consistently yielding results aligned with Lloyd's algorithm.

C. Experiments with real data

C.1. Datasets

We evaluate our approach on six well-established benchmark datasets taken from `Gluonts` (Alexandrov et al., 2020), each containing strictly positive and bounded real-valued series. See Lai et al. (2018) for an extensive description.

- The SOLAR dataset (Lai et al., 2018) consists of hourly aggregated solar power outputs from 137 photovoltaic sites, spanning 7009 time steps. Strong daily seasonality is typically observed due to the day–night cycle.
- The ELECTRICITY dataset (Asuncion et al., 2007) contains hourly power consumption data from 370 clients across 5833 time steps. Demand patterns often exhibit both daily and weekly periodicities, driven by regular human activity and business operations.
- The EXCHANGE dataset (Lai et al., 2018) features daily exchange rates for eight different currencies, with 6071 observations per series. Unlike energy or traffic data, exchange rates often lack clear seasonal patterns and are influenced by broader macroeconomic factors.
- The TRAFFIC dataset (Asuncion et al., 2007) comprises occupancy rates (ranging from 0 to 1) measured hourly by 963 road sensors over 4001 time steps. These series generally display recurrent rush-hour peaks as well as differences between weekdays and weekends.
- The TAXI dataset consists of traffic time-series data of New York City taxi rides, recorded at 1214 locations every 30 minutes in January 2015 (training set) and January 2016 (test set). We used the preprocessed version from (Salinas et al., 2019).
- The WIKIPEDIA dataset (Gasthaus et al., 2019) contains daily views of 2000 Wikipedia pages.

Table 5 provides an overview of the main characteristics of these six datasets. Note for each dataset, we used the official train/test split. We dedicate 10 times the number of prediction steps for validation, at the end of the training data.

C.2. Metrics

In all the following, each model produces K hypotheses $\{\hat{x}^k_{t_0:T}\}_k$. Additionally `Time-MCL` predicts K scores (one per trajectory), that we denote as $\hat{\gamma}_1, \dots, \hat{\gamma}_K \in [0,1]$, with $\sum_{k=1}^{K} \hat{\gamma}_k = 1$, omitting the input $x_{1:t_0-1}$ for conciseness. The observed trajectory is denoted as $x_{t_0:T} \in \mathbb{R}^{D \times (T-t_0+1)}$. We used the `Gluonts` integrated `Evaluator` to compute the

Table 5. **Datasets characteristics.**

DATASET	DIMENSION D	DOMAIN $\mathcal{X}$	FREQ.	TIME STEPS	PRED STEPS $T - t_0 + 1$
SOLAR	137	$\mathbb{R}^+$	HOUR	7,009	24
ELECTRICITY	370	$\mathbb{R}^+$	HOUR	5,833	24
EXCHANGE	8	$\mathbb{R}^+$	DAY	6,071	30
TRAFFIC	963	$(0, 1)$	HOUR	4,001	24
TAXI	1,214	$\mathbb{N}$	30-MIN	1,488	24
WIKIPEDIA	2,000	$\mathbb{N}$	DAY	792	30

metrics, and we customized it to include the distortion. For evaluation over the whole test set, we have N input-targets pairs $(x^i_{1:t_0-1}, x^i_{t_0:T})$, $i \in \{1, \ldots, N\}$. We denote by $\hat{x}^{k,i}_t$ the k-th hypothesis generated for input $x^i_{1:t_0-1}$ at time t.

In the following, we define the *summation over dimension* operator as $\mathcal{A} : x \in \mathbb{R}^D \mapsto \sum_{d=1}^D x^d \in \mathbb{R}$, where x^d denotes the d-th dimension of the vector x.

C.2.1. RMSE (ROOT MEAN SQUARE ERROR)

Let us denote by $\bar{x} \in \mathbb{R}^{D\times(T-t_0+1)}$ the conditional mean estimator of $p(x_{t_0:T} \mid x_{1:t_0-1})$ given the probabilistic model we are evaluating. For `TimeMCL`, we used $\bar{x} = \sum_{k=1}^K \hat{\gamma}_k \hat{x}^k_{t_0:T}$ and for the other methods that do not use score heads, $\bar{x} = \frac{1}{K}\sum_{k=1}^K \hat{x}^k_{t_0:T}$. RMSE is defined as:

$$\text{RMSE} \triangleq \sqrt{\frac{1}{N}\sum_{i=1}^N \frac{1}{T - t_0 + 1}\sum_{t=t_0}^T \big(\mathcal{A}(x^i_t) - \mathcal{A}(\bar{x}^i_t)\big)^2} \, . \tag{18}$$

In `Gluonts`, we can access this metric under the name `m_sum_rmse`. The prefix `m_sum` indicates in (18) that the RMSE is computed after aggregating the target and the prediction dimension by dimension.

C.2.2. CRPS (CONTINUOUS RANKED PROBABILITY SCORE)

In the following, we denote by $Q_X(q)$ the quantile of order q a real random variable X, with $q \in [0, 1]$ (when it exists).

Let us define the random variable X^i_t which takes the value $\mathcal{A}(\hat{x}^{k,i}_t)$ with probability $\hat{\gamma}_k$, or uniformly with probability $\frac{1}{K}$ when using no score heads, respectively. Let us introduce:

$$\mathscr{L}_q\big(x^i_{t_0:T}, \hat{x}^i_{t_0:T}\big) \triangleq 2\sum_{t=t_0}^T \left|\mathcal{A}(x^i_t) - Q_{X^i_t}(q)\right|\Big(\mathbb{1}(\mathcal{A}(x^i_t) \leq Q_{X^i_t}(q)) - q\Big), \quad \mathscr{T}(x_{t_0:T}) \triangleq \sum_{i=1}^N\sum_{t=t_0}^T \left|\mathcal{A}(x^i_t)\right|,$$

where $\mathscr{L}_q$ and $\mathscr{T}$ are referred as the `quantileLoss` and `abs_target_sum` in `Gluonts`.

CRPS (Matheson & Winkler, 1976) is computed as

$$\text{CRPS} \triangleq \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \frac{\sum_{i=1}^N \mathscr{L}_q\big(x^i_{t_0:T}, \hat{x}^i_{t_0:T}\big)}{\mathscr{T}(x_{t_0:T})} \, , \tag{19}$$

where we used $\mathcal{Q} = \{0.05, 0.1, \ldots 0.95\}$. Equation (19) is referred to as `m_sum_mean_wQuantileLoss` in the python library, or CRPS-Sum (Salinas et al., 2019; Ashok et al., 2024) in the related literature.

C.2.3. DISTORSION

Distortion, also referred to as the *Oracle* or *Quantization* error in the literature (Pages & Printems, 2009; Lee et al., 2016; Perera et al., 2024) was defined as

$$D_2 \triangleq \frac{1}{N}\sum_{i=1}^N \min_{k\in\{1,\ldots,K\}} \sqrt{\sum_{d=1}^D \frac{1}{T - t_0 + 1}\sum_{t=t_0}^T (x^{i,d}_t - \hat{x}^{i,k,d}_t)^2} \, .$$

This metric was integrated into the `Evaluator` class from `Gluonts`.

Table 6. **Distortion comparison with 8 Hypotheses.** Results are averaged over four training seeds. **`ETS`**, **`Trf.TempFlow`** and **`Tactis2`**, columns are in gray because they don't share the same backbone as the other baselines. Best scores are in **bold**, and second-best are <u>underlined</u>.

	`ETS`	`Trf.TempFlow`	`Tactis2`	**`TimeGrad`**	**`DeepAR`**	**`TempFlow`**	**`TimeMCL(R.)`**	**`TimeMCL(A.)`**
ELEC.	15132 ± 1662	14160 ± 3370	12557 ± 2405	$\mathbf{10172 \pm 680}$	144832 ± 3081	15822 ± 531	12155 ± 414	<u>11509 ± 703</u>
EXCH.	0.057 ± 0.002	0.066 ± 0.012	$\mathbf{0.031 \pm 0.002}$	0.041 ± 0.006	0.062 ± 0.001	0.051 ± 0.007	<u>0.039 ± 0.001</u>	0.042 ± 0.005
SOLAR	654.48 ± 3.45	387.15 ± 18.57	367.54 ± 35.4	371.97 ± 33.31	770.48 ± 19.91	379.93 ± 22.38	$\mathbf{305.63 \pm 28.24}$	<u>347.82 ± 28.82</u>
TRAFFIC	2.65 ± 0.0	1.28 ± 0.02	0.85 ± 0.04	0.79 ± 0.05	2.14 ± 0.06	1.23 ± 0.04	$\mathbf{0.69 \pm 0.02}$	<u>0.71 ± 0.01</u>
TAXI	589.15 ± 0.75	284.25 ± 13.94	245.09 ± 8.79	<u>211.71 ± 4.02</u>	429.01 ± 14.88	272.84 ± 12.14	$\mathbf{187.04 \pm 2.45}$	256.16 ± 24.95
WIKI	722166 ± 5248	528546 ± 11657	$\mathbf{255328 \pm 3386}$	261463 ± 1047	371035 ± 6578	388511 ± 2472	271878 ± 17358	<u>258468 ± 3108</u>

C.2.4. FLOPs (Floating Point Operations)

FLOPs serve as an approximate measure of the model's computational load. For each baseline, we only considered FLOPs evaluation at inference, with a single batch of size 1 on the EXCHANGE dataset. We used the `FlopCountAnalysis` function from the fvcore library to measure it.

C.2.5. Inference Run-time

Inference time was computed on a single NVIDIA GeForce RTX 2080 Ti, while making sure it is the only process that runs on the node. It was measured over the whole test set of EXCHANGE (`make_evaluation_predictions` function in `Gluonts`), using a batch size of 64. For fair and accurate time comparison with respect to the number of hypotheses, we disable parallel sampling for each baseline (by setting `num_parallel_samples` to 1).

C.2.6. Total Variation

We quantify the average smoothness of the sampled trajectories using the Total Variation (TV). Given K sampled trajectories, we defined it as

$$\mathrm{TV} \triangleq \frac{1}{N} \sum_{i=1}^{N} \left(\frac{1}{K} \sum_{k=1}^{K} \sum_{t=t_0}^{T} \left\| \hat{x}_{t+1}^{k,i} - \hat{x}_{t}^{k,i} \right\|_2 \right) . \tag{20}$$

For `TimeMCL`, we sampled the hypotheses in proportion to their scores for Total Variation computation instead of uniformly as in (20) for the other baselines. The smoother the predictions of a probabilistic estimator, the smaller its total variation.

C.3. Experimental details

C.3.1. Architectures, training, and inference

Recurrent Neural Network Backbone. `Time-MCL`, `TempFlow`, `TimeGrad`, and `DeepAR` share the same RNN backbone, which takes as input a concatenation of feature types. Let B be the batch size, T the time series length, and D_{target} the target dimension. The input includes (i) lagged target values, normalized and shaped as $B \times T \times (n_{\text{lags}} \times D_{\text{target}})$, and (ii) Fourier features of shape $B \times T \times (2 \times N_f)$. The combined input to the RNN thus has shape $B \times T \times ((n_{\text{lags}} \times D_{\text{target}}) + (2 \times N_f))$. Our RNN implementation follows Rasul et al. (2021a) and consists of $L = 2$ layers of Long Short-Term Memory (LSTM) cells, each containing $C = 40$ hidden units. The term N_f, which is associated with Fourier transforms, will be further detailed in the following sections.

Sampling and dataloader The time series is first segmented into past and future windows based on predefined lengths and lead time—sampling slicing points randomly during training and using the last timestamp during prediction (`InstanceSlicer`). After transformation, the dataloader assembles inputs used across all models. It separates past and future targets and adds several structured features: (i) a *Target Dimension Indicator* ($B \times D_{\text{target}}$) uniquely labels each target dimension; (ii) an *Observed Values Indicator* replaces missing target entries with dummies and flags them as observed (1) or missing (0); (iii) a *Padding Mask* marks synthetic padding to ensure uniform sequence lengths, preventing them from influencing training. Additionally, temporal features of shape $B \times T_{\text{pred}} \times 2N_f$ encode periodic patterns using sine and cosine projections, where N_f denotes the number of unique values for a given frequency (*e.g.,* $N_f = 12$ for months).

Transformer Backbone. For the transformer-based version of TempFlow (`Trf.TempFlow`) and for `Tactis2`, we

Table 7. **RMSE (↓) comparison for $K = 16$ hypotheses.**

	ETS	Trf.TempFlow	Tactis2	TimeGrad	DeepAR	TempFlow	TimeMCL (R.)	TimeMCL (A.)
ELEC.	25771 ± 1117	20559 ± 11011	19342 ± 5496	$\mathbf{11597 \pm 931}$	145977 ± 3564	23373 ± 2614	$\underline{18571 \pm 3063}$	19154 ± 2657
EXCH.	0.089 ± 0.01	0.147 ± 0.043	$\mathbf{0.062 \pm 0.007}$	$\underline{0.077 \pm 0.027}$	0.086 ± 0.002	0.095 ± 0.021	0.084 ± 0.002	0.103 ± 0.021
SOLAR	3661.86 ± 28.87	4121.74 ± 243.06	$\underline{3531.02 \pm 439.59}$	$\mathbf{3244.92 \pm 247.16}$	7199.09 ± 107.71	3801.72 ± 122.09	3639.23 ± 606.34	3896.48 ± 468.71
TRAFFIC	15.24 ± 0.09	27.57 ± 0.22	11.54 ± 1.29	$\mathbf{4.44 \pm 1.07}$	39.15 ± 1.48	26.77 ± 1.29	$\underline{5.48 \pm 0.71}$	5.73 ± 0.7
TAXI	9779.81 ± 13.42	4223.88 ± 561.11	$\underline{2662.52 \pm 58.79}$	$\mathbf{2357.85 \pm 133.05}$	7323.32 ± 245.35	4436.39 ± 581.06	6855.32 ± 3456.01	8549.15 ± 2691.98
WIKI	765663 ± 39186	1152828 ± 171868	$\underline{564176 \pm 115593}$	$\mathbf{505869 \pm 60432}$	4469007 ± 1015212	864607 ± 105223	1210500 ± 412274	1264339 ± 166347

Table 8. **CRPS-Sum (↓) comparison for $K = 16$ hypotheses.** Results averaged over four training seeds.

	ETS	Trf.TempFlow	Tactis2	TimeGrad	DeepAR	TempFlow	TimeMCL (R.)	TimeMCL (A.)
ELEC.	0.0601 ± 0.0045	0.0434 ± 0.0214	$\underline{0.0409 \pm 0.0105}$	$\mathbf{0.0235 \pm 0.0018}$	0.4234 ± 0.0101	0.0503 ± 0.0038	0.0488 ± 0.0074	0.0481 ± 0.0086
EXCH.	0.0079 ± 0.001	0.0138 ± 0.0039	$\mathbf{0.0066 \pm 0.0014}$	$\underline{0.0074 \pm 0.0029}$	0.0081 ± 0.0002	0.0091 ± 0.0019	0.0103 ± 0.0002	0.0131 ± 0.0031
SOLAR	0.509 ± 0.0041	0.4528 ± 0.0271	0.3919 ± 0.0378	$\mathbf{0.3789 \pm 0.0356}$	1.028 ± 0.0225	0.4352 ± 0.0215	$\underline{0.3831 \pm 0.0658}$	0.3966 ± 0.0079
TRAFFIC	0.2007 ± 0.0014	0.394 ± 0.0092	0.1339 ± 0.0166	$\mathbf{0.0561 \pm 0.0137}$	0.5257 ± 0.0205	0.3833 ± 0.0271	0.067 ± 0.0102	$\underline{0.0653 \pm 0.009}$
TAXI	0.8948 ± 0.0017	0.288 ± 0.0467	$\underline{0.1756 \pm 0.0056}$	$\mathbf{0.1449 \pm 0.0118}$	0.5223 ± 0.0239	0.3106 ± 0.0512	0.4438 ± 0.1965	0.5803 ± 0.2206
WIKI	0.0789 ± 0.0041	0.1309 ± 0.0316	$\underline{0.0649 \pm 0.019}$	$\mathbf{0.0567 \pm 0.0084}$	0.6588 ± 0.2225	0.0905 ± 0.0051	0.142 ± 0.0624	0.1479 ± 0.0392

employed the same architectures as in their respective original implementations (Rasul et al., 2021b; Ashok et al., 2024).

Optimization Training is conducted using the Adam optimizer with a learning rate of 10^{-3}, and following an 'LR on plateau' scheduler (See Pytorch documentation), a weight decay of 10^{-8} and 200 training epochs. Additionally, a separate validation split is used with a temporal size equal to 10 times the prediction length. When training `TimeMCL`, the WTA loss (and its variants) was divided by the prediction length $T - t_0 + 1$. All models are trained using gradient norm clipping with a threshold of 10. Training and evaluation are performed in single-precision (float32) with PyTorch.

Inference. We used the official experimental protocol for evaluation in this benchmark (*e.g.,* (Rasul et al., 2021a)). The official test dataset is divided into multiple non-overlapping subsets, each containing sufficient points for both context and prediction lengths, allowing comprehensive assessment across different temporal segments. To compute the `TimeMCL` metrics while respecting the probabilities given by the scores, we applied a simple trick: after a single forward pass, we obtain K hypotheses along with their associated probabilities (derived from the scores). To ensure that the hypotheses contribute to the metrics in proportion to their assigned probabilities, we performed resampling with replacement from these K hypotheses based on their respective probabilities before computing metrics.

C.3.2. BASELINES

`DeepAR`. (Salinas et al., 2020). The output distribution as a multivariate normal distribution with mean vector $\mu \in \mathbb{R}^D$, a low-rank covariance factor $L \in \mathbb{R}^{D\times r}$, and a diagonal covariance vector $\sigma \in \mathbb{R}^D$, where the covariance matrix Σ is defined as $\Sigma = LL^\top + \mathrm{diag}(\sigma)$, thereby ensuring efficient parameterization and capturing primary dependencies among the target dimensions. In all the experiments, we set the rank r to 1.

`TimeGrad`. (Rasul et al., 2021a). For the diffusion process, as in Rasul et al. (2021a) we employ 100 diffusion steps with a linear variance schedule, transitioning from $\beta_1 = 1 \times 10^{-4}$ to $\beta_{100} = 0.1$. The loss function utilized is the L^2 loss. The residual architecture is integrated within the diffusion component to facilitate effective training and capture complex temporal dependencies. This residual network consists of 8 residual layers, each with 8 residual channels. The dilation cycle length is set to 2, which controls the expansion of the receptive field without increasing the number of parameters.

`TempFlow` Rasul et al. (2021b). TempFlow applies a conditional normalizing flow (Papamakarios et al., 2021) that is invertible and has a tractable Jacobian, yielding an exact likelihood. Batch normalization after each coupling layer and per-series mean scaling further stabilizes optimization. We adopt the original `TempFlow` configuration from Rasul et al. (2021b): a two-layer LSTM conditioner with 40 cells and dropout 0.1, fed with a 100-step context, and a Real-valued Non-Volume Preserving (RealNVP) (Dinh et al., 2017) head composed of 3 coupling blocks whose scale/shift networks have 2 hidden layers with 100 hidden units. The transformers version of TempFlow `Trf.TempFlow` uses 8 heads with 3 encoder and 3 decoding layers, with an embedding dimension of 32, and a feedforward dimension of 128 with a dropout rate of 0.1 and Gelu (Hendrycks & Gimpel, 2016) activation layers.

Table 9. **Distorsion (↓) Comparison for Solar Dataset.** Results averaged over four training seeds. In this table, the distortion is computed with a variable number of hypotheses K for each baseline, as in Table 4 of the main paper.

K	`ETS`	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL(R.)`	`TimeMCL(A.)`
1	685.77 ± 23.08	468.84 ± 32.54	433.76 ± 19.02	**398.96 ± 32.38**	838.14 ± 26.13	422.24 ± 10.64	422.75 ± 84.28	422.75 ± 84.28
2	678.49 ± 13.29	424.7 ± 14.47	410.23 ± 12.98	**380.99 ± 31.06**	817.75 ± 15.28	404.89 ± 20.53	384.28 ± 19.92	418.27 ± 95.95
3	678.94 ± 10.6	409.97 ± 20.37	377.71 ± 34.55	377.1 ± 30.87	794.45 ± 19.39	398.96 ± 23.97	**356.42 ± 44.11**	358.48 ± 20.29
4	671.97 ± 16.52	402.49 ± 20.51	376.91 ± 34.72	375.25 ± 32.23	789.66 ± 18.97	386.28 ± 18.5	**326.62 ± 20.89**	347.45 ± 14.59
5	666.44 ± 6.8	400.63 ± 18.4	374.76 ± 32.31	374.96 ± 32.36	776.94 ± 17.93	385.54 ± 19.14	376.25 ± 43.23	**323.2 ± 18.3**
8	654.48 ± 3.45	387.15 ± 18.57	367.54 ± 35.4	371.97 ± 33.31	770.48 ± 19.91	379.93 ± 22.38	**305.63 ± 28.24**	347.82 ± 28.82
16	641.32 ± 3.57	374.69 ± 20.97	358.01 ± 38.47	360.6 ± 34.79	748.68 ± 18.92	371.14 ± 18.2	**280.03 ± 15.22**	305.5 ± 4.05

`Tactis2` (Ashok et al., 2024). Unlike the RNN-based `TimeGrad`, `TempFlow`, and `Time-MCL`, `Tactis2` encodes each time step as a token that includes the value, a binary missingness flag, static covariates, and a positional code. Two Transformer encoders process these tokens to produce separate embeddings: one for the marginal distributions and another for the dependence structure. A hyper-network transforms the marginal embedding into the parameters of a Deep-Sigmoidal Flow, which maps the observation to its probability integral transform. An *attentional copula* conditioned on the dependence embedding captures cross-series interactions. The training follows a two-stage curriculum: first, learning the marginals associated with each dimension; then freezing them and fitting the copula. We use the default hyperparameters reported by Ashok et al. (2024) for `fred-md` (Godahewa et al., 2021), detailed in this notebook from the official code. Our setup uses 5-dimensional time-series embeddings. Both the Marginal CDF Encoder and the Attentional Copulas Encoder have two layers with one attention head (dimension 16, no dropout). The Transformer Decoder has one attention layer (dimension 8) with 3 heads, a 2-layer MLP (hidden dimension 48), and 20 histogram bins. The Deep Sigmoidal Flow marginals (Huang et al., 2018) consist of 2 flow layers of dimension 8 and a 2-layer MLP (hidden dimension 48). The training comprises 20 epochs in phase 1 and 180 epochs in phase 2. Each time series is standardized by subtracting its training-set mean and dividing by its training-set standard deviation, ensuring zero mean and unit variance before model ingestion. Optimization parameters are kept the same as in the other baselines.

`ETS`. Exponential smoothing (Hyndman et al., 2008) is a non-neural model. Holt-Winters Exponential Smoothing was applied with an additive trend and an additive seasonality, assuming a seasonal period of 24. Model parameters were estimated through automatic optimization using the `statsmodels` library.

`TimeMCL (Relaxed)` The Relaxed-WTA Loss (Rupprecht et al., 2017) was computed for each pair $(x_{1:t_0-1}, x_{t_0:T})$ as

$$(1-\varepsilon)\,\mathcal{L}_\theta^{k^\star}(x_{1:t_0-1}, x_{t_0:T}) + \frac{\varepsilon}{K-1}\sum_{s=1,\, s\neq k^\star}^{K} \mathcal{L}_\theta^{s}(x_{1:t_0-1}, x_{t_0:T})\;, \tag{21}$$

with $\varepsilon = 0.1$. We found that this provides a good trade-off for handling under-trained hypotheses without deteriorating the distortion performance. However, we did not specifically tune this value for each dataset.

`TimeMCL (Annealed)` The annealed MCL (aMCL) Loss (Perera et al., 2024) was computed with an exponential temperature scheduler; $T(t) = T_0\rho^t$, where t is the number of the training epoch, T_0 is the initial temperature, and ρ is the decay factor. At each temperature T, the aMCL loss is computed for each pair $(x_{1:t_0-1}, x_{t_0:T})$ as

$$\sum_{k=1}^{K} q_k(T)\mathcal{L}_\theta^{k}(x_{1:t_0-1}, x_{t_0:T})\;, \tag{22}$$

where the coefficients

$$q_k(T) \triangleq \frac{1}{Z(x_{1:t_0-1}, x_{t_0:T}; T)}\exp\left(-\frac{\mathcal{L}_\theta^{k}(x_{1:t_0-1}, x_{t_0:T})}{T}\right), \quad Z(x_{1:t_0-1}, x_{t_0:T}; T) \triangleq \sum_{s=1}^{K}\exp\left(-\frac{\ell(\mathcal{L}_\theta^{s}(x_{1:t_0-1}, x_{t_0:T})}{T}\right), \tag{23}$$

are detached from the computational graph. In our experiments with aMCL, we set $\rho = 0.95$ and $T_0 = 10$, and we set as limit temperature $T_{\lim} = 5\times10^{-4}$ before switching back to the vanilla WTA mode.

Table 10. **Inference time (in seconds) for EXCHANGE dataset.** Results averaged over 15 random seeds. For accurate time comparison with respect to K, we disable parallel computation of the samples for each baseline. `ETS`, which doesn't require neural networks is also included for completeness (in gray). See Appendix C.2.5 for details.

K	`ETS`	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL`
1	0.06 ± 0.01	0.19 ± 0.02	0.56 ± 0.04	14.92 ± 0.23	0.09 ± 0.01	0.13 ± 0.01	**0.08 ± 0.01**
2	0.06 ± 0.00	0.34 ± 0.01	1.10 ± 0.05	30.04 ± 0.49	0.13 ± 0.01	0.22 ± 0.01	**0.11 ± 0.01**
3	0.06 ± 0.00	0.48 ± 0.02	1.63 ± 0.07	45.10 ± 0.50	0.18 ± 0.01	0.30 ± 0.02	**0.14 ± 0.01**
4	0.06 ± 0.00	0.63 ± 0.02	2.18 ± 0.08	60.07 ± 0.97	0.22 ± 0.01	0.39 ± 0.01	**0.18 ± 0.01**
5	0.07 ± 0.00	0.79 ± 0.04	2.75 ± 0.20	75.69 ± 0.86	0.26 ± 0.01	0.49 ± 0.05	**0.23 ± 0.02**
8	0.06 ± 0.00	1.22 ± 0.05	4.32 ± 0.11	119.67 ± 0.97	**0.38 ± 0.02**	0.73 ± 0.03	0.39 ± 0.01
16	0.06 ± 0.00	2.47 ± 0.23	8.69 ± 0.36	241.57 ± 2.24	**0.70 ± 0.04**	1.39 ± 0.03	1.12 ± 0.04

Table 11. **Total Variation (↓) comparison for $K = 16$ hypotheses. `ETS`, `Trf.TempFlow` and `Tactis2`** columns are in gray because they don't share the same backbone as the other baseline. Best methods are **bold**, second best are underlined. Results averaged over four training seeds. Total Variation quantifies the average smoothness of the predicted trajectories (lower is more smooth).

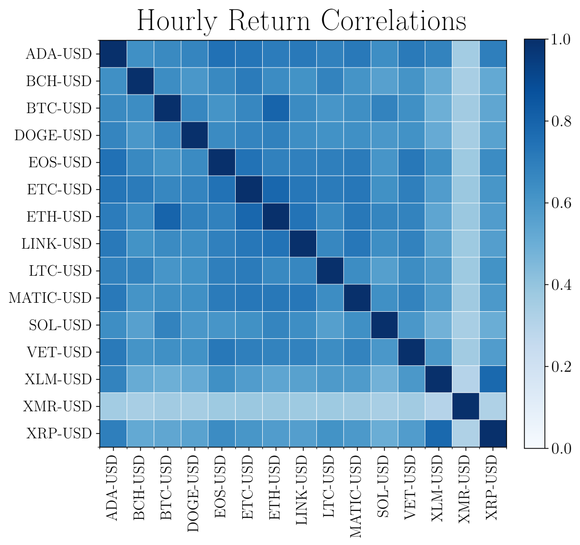
	`ETS`	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL(R.)`	`TimeMCL(A.)`
ELEC.	315611 ± 1858	451750 ± 61031	284232 ± 23269	372443 ± 18499	4584308 ± 97542	434097 ± 34623	**220375 ± 33961**	245908 ± 30505
EXCH.	0.426 ± 0.0052	0.633 ± 0.0764	0.237 ± 0.0355	0.606 ± 0.0208	2.075 ± 0.0434	1.104 ± 0.1582	**0.031 ± 0.0075**	0.042 ± 0.0141
SOLAR	5899 ± 17	3924 ± 381	4421 ± 828	5383 ± 2173	10163 ± 229	3653 ± 488	3391 ± 947	**2195 ± 297**
TRAFFIC	18.782 ± 0.016	20.56 ± 0.932	11.063 ± 0.874	12.991 ± 1.661	69.603 ± 2.61	18.114 ± 0.512	5.766 ± 0.19	**5.714 ± 0.283**
TAXI	4385.41 ± 11.46	5331.4 ± 517.54	5452.77 ± 324.07	4232.16 ± 713.9	7932.26 ± 711.87	4147.2 ± 362.87	709.86 ± 332.07	**701.61 ± 199.35**
WIKI	19692927 ± 269644	18843511 ± 1553022	2634547 ± 597323	2458503 ± 151593	9983566 ± 434670	12410909 ± 319517	**9532 ± 10690**	271611 ± 359320

Table 12. **Results of neural networks-based methods on the cryptocurrency dataset.** Here, $K = 4$, and the results were averaged over three random seeds. Here, `TimeMCL` follows the same experimental setup as in the previous benchmark, except that we used Z-Score normalization (instead of mean scaling) during training.

	`Trf.TempFlow`	`Tactis2`	`TimeGrad`	`DeepAR`	`TempFlow`	`TimeMCL(A.)`
Distortion	**1334.441 ± 245.746**	1898.896 ± 255.785	1834.202 ± 188.147	2437.798 ± 94.987	1870.915 ± 174.434	1400.396 ± 144.5
Total Variation	8895.81 ± 1198.262	4209.638 ± 1047.498	12797.597 ± 3040.59	32352.73 ± 3054.369	9957.855 ± 2982.198	**2174.019 ± 769.041**
CRPS	**0.014 ± 0.001**	0.018 ± 0.001	0.018 ± 0.003	0.019 ± 0.001	0.02 ± 0.001	0.016 ± 0.004
RMSE	**2275.986 ± 165.877**	2515.755 ± 138.424	2642.089 ± 366.955	2743.841 ± 93.297	2756.95 ± 68.081	2528.18 ± 687.728
FLOPs	3.89×10^7	9.81×10^7	9.13×10^8	**1.74×10^5**	1.94×10^7	9.98×10^5

C.4. Evaluation on financial data

An application of **`Time-MCL`** is in forecasting financial time series, such as asset prices. Rather than working directly with raw prices, it is common practice to use the log returns (Tsay, 2005), defined as $x_t^d = \log P_t^d - \log P_{t-1}^d$, where P_t^d represents the price of asset d at time t. These log returns are challenging to forecast. Capturing the extreme tails of their distribution can yield particularly valuable insights for financial applications.



Ticker	Min.	Max.	Points.
ADA	0.24	1.31	14 614
BCH	180.45	711.92	14 614
BTC	25 001.41	108 240.09	14 614
DOGE	0.06	0.48	14 607
EOS	0.40	1.50	14 614
ETC	14.52	39.67	14 614
ETH	1 528.68	4 076.34	14 613
LINK	5.77	30.44	14 614
LTC	51.16	144.77	14 607
MATIC	0.26	1.28	14 607
SOL	17.49	287.81	14 613
VET	0.01	0.08	14 614
XLM	0.08	0.60	14 614
XMR	101.70	241.71	14 614
XRP	0.40	3.39	14 614

Figure 5. **Cryptocurrency dataset description.** *(Left)* Pair-wise matrix correlations of log returns between the time series. *(Right)* Yahoo Finance ticker symbols, number of hourly observations per asset (**Points**), and the corresponding price scales (**Min, Max**), thereby summarizing both data volume and cross-asset dependence.

We obtained cryptocurrency data from `YahooFinance` and was divided into three splits: training (2023-07-01 to 2024-09-01), validation (2024-09-02 to 2024-12-05), and test (2024-12-06 to 2025-03-01), with an hourly resolution. Figure 5 provides an overview of the assets collected for training, including return correlations among these assets, their price ranges, and the number of data points available. Missing data points were handled using forward filling.

Setup. We trained and evaluated the previously introduced neural-based baselines on this dataset, using $K = 4$ hypotheses. Each model was trained for 100 epochs, with 100 iterations per epoch and a batch size of 64. The prediction length and context length were both set to 24. The experimental setup was the same as in the previous benchmark, except that we applied Z-score normalization instead of mean scaling during training, which better handles the wide range of asset price scales. With the exception of `Tactis2`, which already uses Z-score normalization, we observed that this change significantly improved the performance of all baselines, except for `DeepAR` where we retained mean scaling. Note that, because the validation losses were monotonically decreasing, we used the final training checkpoint for testing.

Results. Quantitative results on quality (Distortion, CRPS, RMSE), smoothness (Total Variation), and computational cost (FLOPs) are presented in Table 12. `TimeMCL` produces the smoothest predictions overall. Among non-transformer-based baselines (`TimeGrad`, `DeepAR`, and `TempFlow`), `TimeMCL` consistently outperforms the others in terms of quality. In this setup, it also performs competitively with `Tactis2`, which was a strong competitor in the previous benchmark. Unlike in the previous benchmark, we found that the Transformer-based variant of TempFlow (`Trf.TempFlow`) now outperforms the original `TempFlow` and slightly surpasses `TimeMCL` in quality metrics, though it requires nearly 40 times more FLOPs. In the future, we plan to refine this comparison by implementing a Transformer-based version of `TimeMCL`.

Cryptocurrency price predictions are visualized in Figure 6. Each row represents a target dimension, and each column corresponds to a method listed in Table 12. Following the notation introduced in Figure, `TimeMCL` predictions are shown in shades of blue, with color intensity reflecting the associated score. We observe that `TimeMCL` produces smoother predictions compared to other methods and effectively captures different modes in the conditional distribution. For clarity, only a subset of the cryptocurrencies is shown in the figure; however, all models were trained to jointly predict all cryptocurrencies.

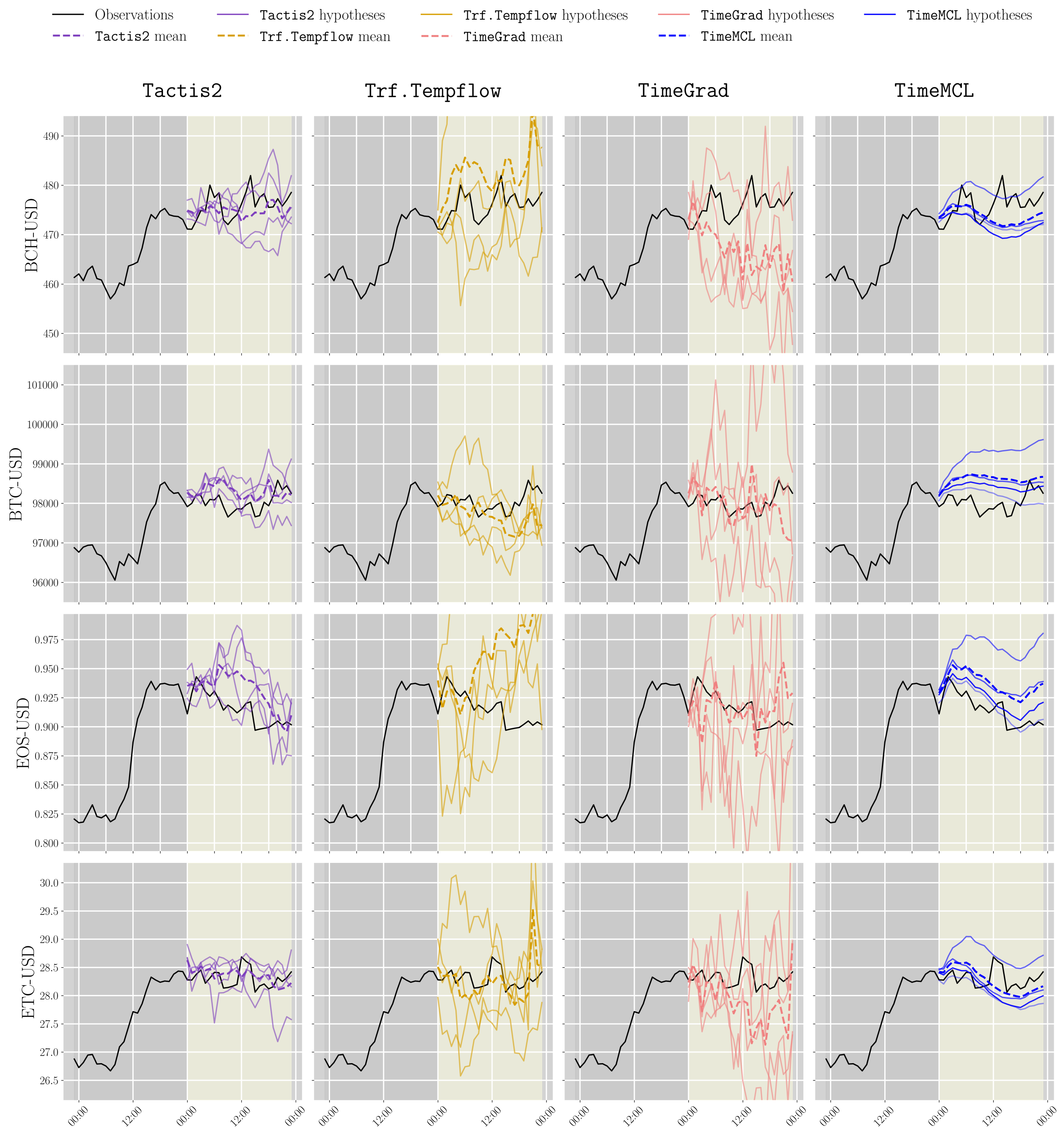


Figure 6. **Cryptocurrency Price Prediction on 04/01/2025 with $K = 4$ Hypotheses.**

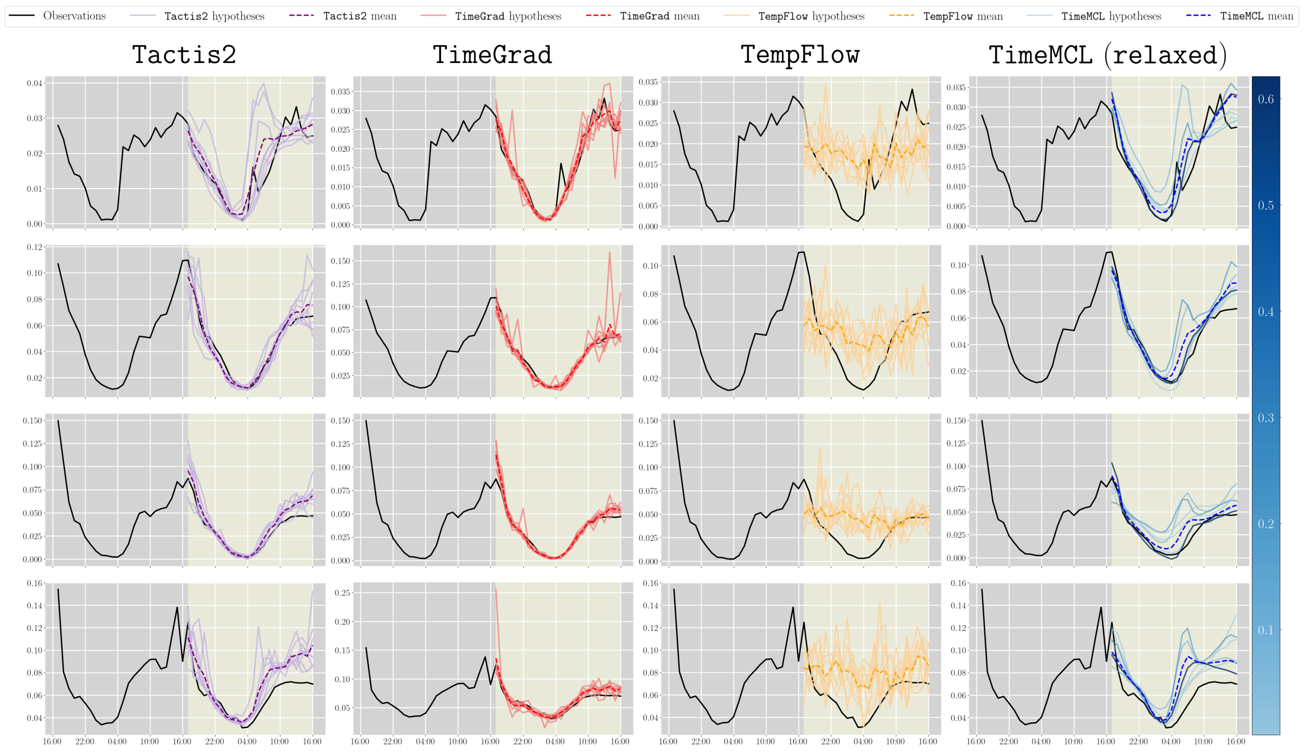


Figure 7. **Qualitative results on the TRAFFIC dataset**, with same experimental setup as in Figure 3 with 8 hypotheses.

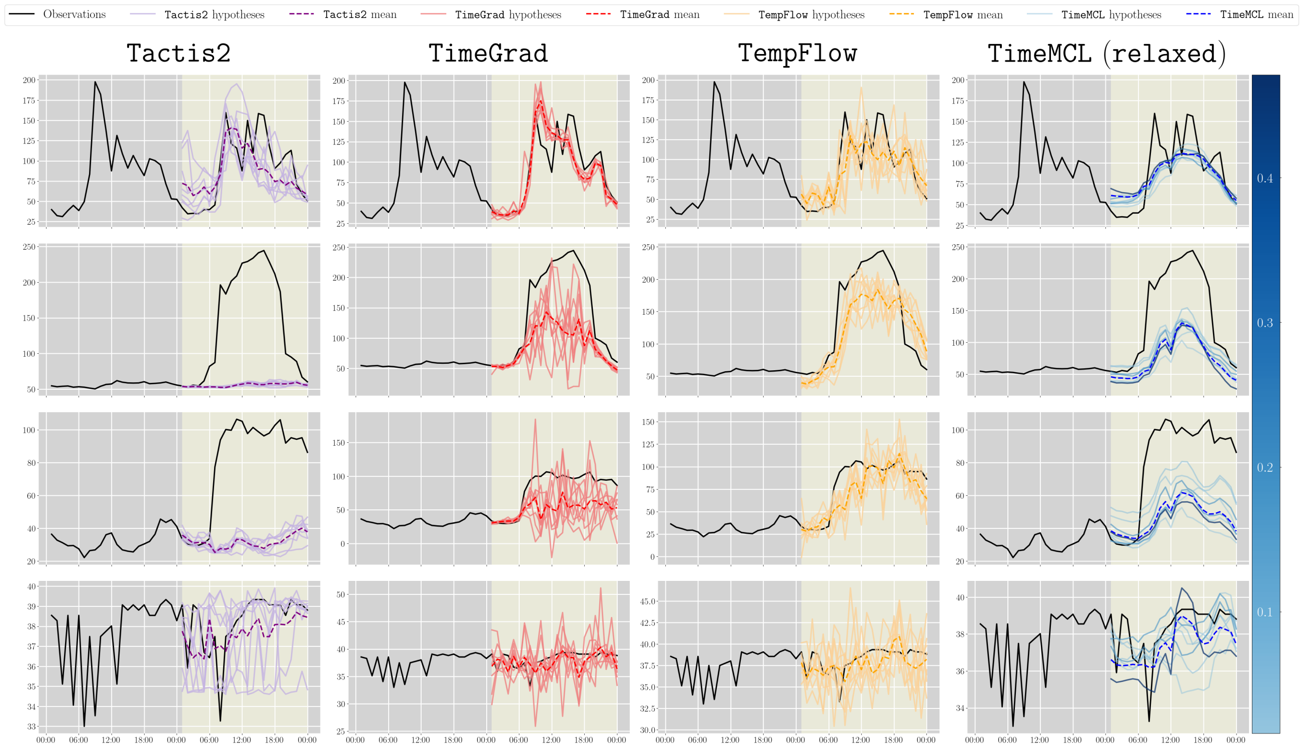


Figure 8. **Qualitative results on the ELECTRICITY dataset**, with same experimental setup as in Figure 3 with 8 hypotheses.