QUANTILE-GUIDED DIFFUSION FOR PROBABILISTIC TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Quantile forecasts summarize uncertainty at each future time step, while diffusion forecasters represent it through sampled future trajectories. We study how predicted quantiles can serve as distributional controls for diffusion forecasting. Specifically, we investigate how the mechanism of quantile integration affects its effectiveness in guiding a pretrained diffusion forecaster. We find that residual control throughout the denoising network yields the most consistent improvements. We instantiate this approach as *Control-TS*, which learns a quantile-control pathway while keeping the pretrained backbone frozen. Across eight benchmarks, Control-TS consistently improves the backbone's CRPS, achieves competitive performance against state-of-the-art forecasting models, and retains its gains in capacity-matched comparisons. Further analyses examine how these gains depend on quantile granularity, predictor choice, and forecast quality. Finally, we introduce the *Conditional Linear Predictive Score (CLPS)*, which complements distributional metrics by evaluating the downstream forecasting utility of generated futures. Together, these findings support predicted quantiles as an effective interface between explicit uncertainty estimates and diffusion-based trajectory generation.

1 INTRODUCTION

Time-series forecasting supports decision-making in domains such as healthcare, retail, renewable energy, finance, and climate (Bui et al., 2018; Mudelsee, 2019; Sezer et al., 2020; Fildes et al., 2022; Ying et al., 2023). Point forecasts alone are often insufficient: decisions also depend on the range and likelihood of possible outcomes (Hyndman & Athanasopoulos, 2021). Probabilistic forecasting addresses this need by characterizing a distribution over possible future outcomes.

Predictive distributions can be represented in various ways. Quantile forecasts provide explicit, horizon-wise summaries of predictive location, dispersion, and asymmetry (Salinas et al., 2020; Lim et al., 2021), while diffusion forecasters represent uncertainty through variation across sampled future trajectories (Biloš et al., 2023; Kollovieh et al., 2023).

Forecasters trained through quantile regression or autoregressive likelihood estimation may capture predictive patterns that complement those learned through diffusion denoising. Their quantile forecasts could make this predictive information available throughout the denoising process. However, these quantiles are imperfect and do not specify dependence across future time steps. The challenge is to exploit these marginal summaries while leaving the diffusion model responsible for generating complete future trajectories. We therefore ask: *Can externally predicted quantile trajectories provide effective guidance for diffusion forecasting, and if so, how should they be incorporated?*

We study this question through *distributional control*: using explicit summaries of a predictive distribution to guide diffusion forecasting. We use TSDiff (Kollovieh et al., 2023) as our primary experimental backbone since it separates joint sequence modeling from forecast conditioning, enabling quantile control of a frozen pretrained denoiser. Starting from the same pretrained backbone and quantile forecasts, we investigate how output-level ensembling and input-, feature-, and residual-level conditioning affect forecasting performance, as summarized in Figure 1. Notably, every evaluated mechanism improves CRPS on some datasets, demonstrating the promise of external predictive information across distinct interfaces. Among these, full-depth residual control, instantiated as *Control-TS*, is the only design to yield improvements across all eight benchmarks. This consistency motivates our focus on Control-TS in the subsequent analyses.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

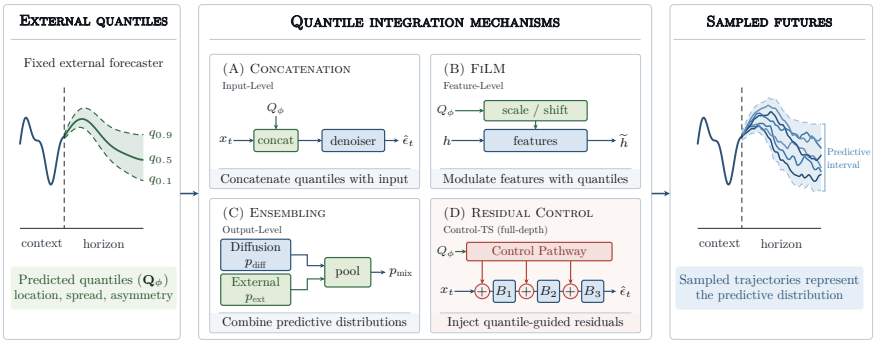


Figure 1: **Integrating predicted quantiles with diffusion forecasting.** (A) concatenates, (B) FiLM, (C) ensembles. (D) adds residuals before the first 1, 2, or all blocks (Control-TS). All improve backbone CRPS on some datasets; full-depth Control-TS gives the most consistent gains on all eight.

Control-TS converts predicted quantile trajectories into learned residual guidance for diffusion forecasting. It uses an auxiliary trainable controller to adapt a frozen denoiser, building on prior work on residual adapters, side-tuning, and spatial conditioning of image diffusion models (Rebuffi et al., 2018; Zhang et al., 2020; Zhang et al., 2023). The controller processes horizon-aligned quantiles and the diffusion timestep, producing residual corrections throughout the denoising network. To isolate the quantiles' contribution, we also evaluate observed history as an alternative control input. Additionally, we examine how quantile granularity, predictor choice, and degraded or misaligned controls affect guidance effectiveness. Analyses of controller design and computational cost characterize the resulting accuracy-cost tradeoff.

Across eight forecasting benchmarks, Control-TS improves CRPS over its diffusion backbone on every dataset, with reductions of up to approximately 35%, and achieves the best or tied-best mean CRPS among all evaluated forecasters on five benchmarks. It also outperforms a capacity-matched diffusion backbone on six datasets and TSDiff's training-free refinement of external forecasts on seven, supporting benefits beyond increased capacity or access to external forecasts alone. Experiments with CSDI (Tashiro et al., 2021) and MG-TSD (Fan et al., 2024) further demonstrate the applicability of quantile control to other diffusion forecasters, with gains varying across backbones and datasets.

To complement distributional evaluation, we ask whether generated futures can serve as useful training targets for downstream forecasting. Horizon-wise CRPS measures marginal predictive accuracy (Gneiting & Raftery, 2007; Hersbach, 2000), but does not directly assess this utility. We introduce the *Conditional Linear Predictive Score (CLPS)*, a Train-on-Hybrid, Test-on-Real (THTR) criterion (Esteban et al., 2017) that trains a linear predictor on real contexts paired with generated futures and evaluates it on held-out real data. The shared linear protocol is computationally inexpensive, with deterministic fitting for fixed data. Lower scores indicate greater downstream utility under this protocol. Control-TS improves CLPS over its backbone on six of eight benchmarks.

Contributions. [1]

1. **Quantile-guided diffusion forecasting.** We establish predicted quantile trajectories as effective distributional controls through a systematic study of interaction mechanisms, quantile granularity, predictor choice, and control quality. Full-depth residual control is the most consistently effective interface on our primary backbone.
2. **Control-TS and empirical validation.** We instantiate this interface as Control-TS, improving the primary backbone on all eight benchmarks, outperforming a capacity-matched diffusion baseline on six, and achieving best or tied-best overall CRPS on five. Further experiments assess benefits across backbones, controller design, and computational tradeoffs.
3. **Complementary predictive evaluation.** We introduce CLPS to assess generated futures' forecasting utility via a shared linear evaluator with inexpensive, deterministic fitting for fixed data. Control-TS outperforms its diffusion backbone on six of eight benchmarks.

[1] Code is available at https://anonymous.4open.science/r/quantile-guided-diffusion-anonymous-7BEC. The code will be released publicly upon acceptance.

2 QUANTILE CONTROL FOR DIFFUSION FORECASTING

2.1 QUANTILE FORECASTS AS DISTRIBUTIONAL CONTROLS

We consider univariate probabilistic forecasting. Let $\mathbf{y}_{\mathrm{ctx}} \in \mathbb{R}^{L_{\mathrm{ctx}}}$ denote the observed context and $\mathbf{Y}_{\mathrm{tar}} = (Y_1, \ldots, Y_H)$ the future trajectory, where $H = L_{\mathrm{tar}}$. Our objective is to approximate the unknown conditional distribution $p^\star(\mathbf{Y}_{\mathrm{tar}} \mid \mathbf{y}_{\mathrm{ctx}})$.

An external probabilistic forecaster with fixed learned parameters ϕ estimates the conditional quantiles of each Y_h given $\mathbf{y}_{\mathrm{ctx}}$. For quantile levels $\mathcal{Q} = \{\tau_1, \ldots, \tau_Q\}$ with $0 < \tau_1 < \cdots < \tau_Q < 1$, we collect these estimates in

$$\mathbf{Q}_\phi(\mathbf{y}_{\mathrm{ctx}}) = \left[\hat{Q}_{\phi,h}(\tau_k \mid \mathbf{y}_{\mathrm{ctx}})\right]_{\substack{h=1,\ldots,H \\ k=1,\ldots,Q}} \in \mathbb{R}^{H \times Q}, \tag{1}$$

where $\hat{Q}_{\phi,h}(\tau_k \mid \mathbf{y}_{\mathrm{ctx}})$ is the predicted τ_k-quantile for future step h. These estimates may be predicted directly or computed from forecast samples; any sampling randomness is suppressed in the notation.

We use $\mathbf{Q}_\phi(\mathbf{y}_{\mathrm{ctx}})$ as a *distributional control*: an explicit set of marginal predictive summaries across the forecast horizon. For example, with levels $\{0.1, 0.5, 0.9\}$, the median describes predicted location, the gap between the outer quantiles describes spread, and their relative distances from the median can indicate asymmetry at these levels. These summaries do not specify dependence across future time steps.

We study how these controls can be incorporated into diffusion forecasting. For mechanisms that condition the denoising network on the quantile matrix, we write the predictive distribution as

$$p_\theta(\mathbf{Y}_{\mathrm{tar}} \mid \mathbf{y}_{\mathrm{ctx}}, \mathbf{Q}_\phi(\mathbf{y}_{\mathrm{ctx}});\ \Psi)\,, \tag{2}$$

where Ψ denotes the integration mechanism and θ collects the parameters of the resulting forecasting model for that mechanism. Output-level ensembling mixes the two predictive distributions directly and does not take $\mathbf{Q}_\phi(\mathbf{y}_{\mathrm{ctx}})$ as a network input.

2.2 QUANTILE INTEGRATION MECHANISMS

We examine output-level ensembling and three mechanisms for incorporating quantile controls within the denoising network, illustrated in Figure 1.

Output-level ensembling. We form a linear pool of the diffusion and external predictive distributions, with the mixing weight selected by validation CRPS. Both forecasters remain unchanged, and their forecasts are combined at the output without modifying the denoising process.

Input-level conditioning. We concatenate the quantile trajectories with the noisy denoiser input at each diffusion step. Quantile information enters through the input layer and propagates through the backbone's subsequent representations. The reported variant fine-tunes the backbone so its weights can adapt to the added quantile channels.

Feature-level modulation. Feature-wise linear modulation (FiLM) (Perez et al., 2018) maps the quantile controls to scale and shift parameters that modulate intermediate denoiser features. We train the modulation parameters and fine-tune the backbone to adapt the denoiser to the altered intermediate representations.

Residual control. At each diffusion step, a trainable temporal controller processes the quantile trajectories conditioned on the timestep. Its block outputs pass through zero-initialized 1×1 projections and are added to the inputs of the corresponding residual blocks in the frozen backbone. We compare injection before the first block, the first two blocks, or all residual blocks, yielding one-block, two-block, and full-depth variants, respectively.

Section 3.1 compares these mechanisms. The following subsection describes Control-TS, our implementation of full-depth residual control.

2.3 CONTROL-TS: RESIDUAL QUANTILE CONTROL

We instantiate full-depth residual control as Control-TS on TSDiff (Kollovieh et al., 2023), an unconditional diffusion model whose architecture builds on SSSD (Lopez Alcaraz & Strodthoff,

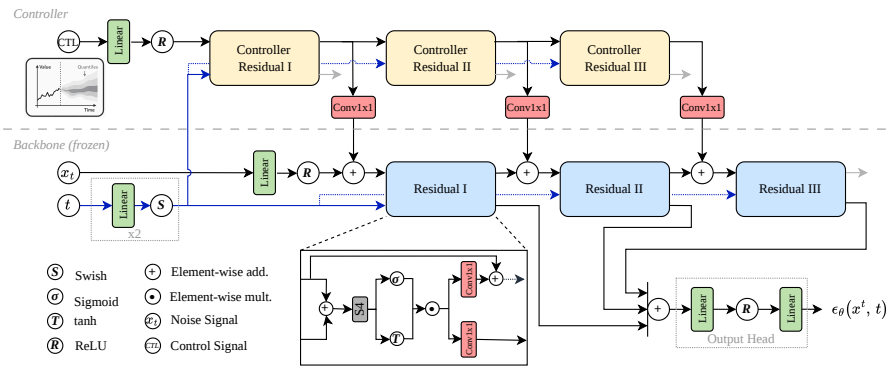


Figure 2: **Control-TS, the full-depth residual realization.** A dedicated quantile pathway injects depth-aligned control through zero-initialized 1×1 projections into the frozen backbone.

2023), which adapts DiffWave (Kong et al., 2021) using S4 temporal blocks (Gu et al., 2022). For Control-TS, we fix Ψ to full-depth residual control and $\theta = (\theta_B, \theta_C)$, where θ_B denotes the pretrained diffusion-backbone parameters and θ_C denotes the control-pathway parameters, including its input projection and zero-initialized residual projections. TSDiff learns a noise predictor $\epsilon_{\theta_B}(x_t, t)$, where x_t denotes the noisy context-future sequence at diffusion step t. Forecasting conditions reverse diffusion on the observed context through observation self-guidance (Kollovieh et al., 2023). We freeze θ_B and optimize only θ_C. Full DDPM and guidance equations are given in Appendix A.1.

Control-TS separates generation from control: a dedicated pathway learns how predicted quantiles should influence the frozen backbone at every residual block (Figure 2).

The controller comprises three randomly initialized residual blocks matching the backbone's residual-stack architecture (Appendix A.2). Let $B^{(k)}_{\theta_B}$ and $C^{(k)}_{\theta_C}$ denote the corresponding backbone and Controller blocks, $Z^{(k)}_{\theta_C}$ zero-initialized 1×1 projections (Nichol & Dhariwal, 2021; Zhang et al., 2023), and $\tau(t)$ the timestep embedding. The input projections E_{θ_B} and E_{θ_C} map the noisy sequence x_t and control signal c to their respective hidden representations. With initial states $h^{(0)}_t = E_{\theta_B}(x_t)$ and $u^{(0)}_t = E_{\theta_C}(c)$, the block updates for $k = 1, 2, 3$ are

$$\begin{aligned} u^{(k)}_t &= C^{(k)}_{\theta_C}\left(u^{(k-1)}_t, \tau(t)\right), \\ h^{(k)}_t &= B^{(k)}_{\theta_B}\left(h^{(k-1)}_t + Z^{(k)}_{\theta_C}(u^{(k)}_t), \tau(t)\right). \end{aligned} \tag{3}$$

Because every $Z^{(k)}_{\theta_C}$ initially outputs zero, the controlled noise predictor exactly matches the pretrained backbone's noise predictor at initialization. Section 3.1 evaluates the contributions of zero initialization and full-depth control.

2.4 Training and Forecast Generation

We instantiate the external forecaster with DeepAR (Salinas et al., 2020) or TFT (Lim et al., 2021) in GluonTS (Alexandrov et al., 2020). Q1 uses $\mathcal{Q} = \{0.5\}$, Q3 uses $\{0.1, 0.5, 0.9\}$, and Q5 uses $\{0.1, 0.25, 0.5, 0.75, 0.9\}$. To align control with the full diffusion sequence, we replicate the scaled observed context across the Q channels and append the predicted horizon quantiles, forming $c \in \mathbb{R}^{(L_{\text{ctx}}+H)\times Q}$. When lag features are enabled, the same lag expansion is applied to the diffusion input and control. Appendices A.3 and A.4 give extraction, scaling, alignment, and lag details.

During training, only θ_C is optimized under the standard diffusion noise-prediction objective; the backbone and external forecaster remain fixed. At inference, independent Gaussian draws are propagated through controlled reverse diffusion, with the original backbone-only observation self-guidance enforcing context consistency. Algorithms 1 and 2 (Appendix A.7) provide complete pseudocode; Appendices A.5 and A.6 show training curves.

3 EXPERIMENTS

We first compare quantile integration mechanisms and examine the design of residual control. We then evaluate Control-TS against established probabilistic forecasters, assess its gains under comparable parameter budgets, and test its applicability to CSDI and MG-TSD. Next, we investigate the role of the control information by comparing predicted quantiles with observed-context controls and varying quantile granularity and predictor choice. We subsequently assess robustness to degraded controls, the downstream forecasting utility of generated futures, and computational cost.

Unless otherwise stated, *Control-TS* denotes full-depth residual control with the external forecaster (DeepAR or TFT) and quantile set (Q1, Q3, or Q5) selected by validation CRPS on each dataset. Complete configuration-level results are provided in Appendix B.10. Appendix B provides full experimental details, with the protocols and hyperparameters specified in Appendix B.4.

Datasets and evaluation. Our main experiments evaluate *univariate probabilistic forecasting* on eight real-world benchmarks from the GluonTS repository (Alexandrov et al., 2020): Electricity (Dua & Graff, 2017), Exchange (Lai et al., 2018), KDDCup (ACM SIGKDD, 2018; Godahewa et al., 2020), M4-Hourly (Makridakis et al., 2020), Solar (Lai et al., 2018), Traffic (Dua & Graff, 2017), UberTLC (FiveThirtyEight, 2016), and Wikipedia (Gasthaus et al., 2019). Dataset statistics, splits, and licenses are documented in Appendices B.1 and B.2. We evaluate probabilistic accuracy using the continuous ranked probability score (CRPS) (Gneiting & Raftery, 2007; Hersbach, 2000), computed from 100 sampled forecast paths; lower values indicate better performance. Appendix B.3 provides the scoring and sampling details. Unless otherwise stated, we report the mean and standard deviation over three independent runs with different random seeds.

3.1 EFFECT OF THE INTEGRATION MECHANISM

We compare the integration mechanisms in Section 2.2, holding the external forecaster fixed within each dataset and using identical quantile levels across quantile-conditioned variants. Concatenation and FiLM fine-tune the backbone, whereas residual variants keep it frozen. The ensembling weight is selected by validation CRPS. We vary residual depth by injecting corrections before the first block, the first two blocks, or all three blocks (Control-TS). We report fine-tuned variants of concatenation and FiLM, allowing the denoiser to adapt to the introduced quantile conditioning. We also evaluated frozen-backbone variants of both mechanisms, but observed worse performance. Full results are in Appendix B.7. TSDiff's training-free refinement (Kollovieh et al., 2023) operates on forecast samples rather than quantile trajectories, so we evaluate it with DeepAR in Section 3.2.

Table 1: **Quantile integration mechanisms.** Best values are **bold**, second-best values are underlined.

Dataset	Backbone	Output ensembling	Input concat.	FiLM	1-block residual	2-block residual	Control-TS (full-depth)
Electricity	0.049±0.000	0.049±0.000	0.062±0.000	0.054±0.000	0.052±0.000	0.052±0.000	**0.047±0.000**
Exchange	0.012±0.001	0.008±0.000	**0.007±0.000**	0.008±0.000	0.009±0.000	0.008±0.000	0.008±0.000
KDDCup	0.312±0.001	0.320±0.003	0.289±0.002	0.268±0.005	0.286±0.001	0.285±0.002	**0.249±0.001**
M4-Hourly	0.034±0.000	0.034±0.000	0.027±0.001	0.026±0.001	0.032±0.000	0.026±0.000	**0.022±0.000**
Solar	0.402±0.000	0.378±0.001	**0.308±0.004**	0.360±0.002	0.444±0.000	0.415±0.001	0.341±0.000
Traffic	0.099±0.000	0.099±0.000	0.117±0.000	0.116±0.001	0.125±0.000	0.118±0.000	**0.097±0.000**
UberTLC	0.172±0.000	**0.163±0.000**	0.187±0.001	0.184±0.001	0.203±0.001	0.193±0.001	0.169±0.000
Wikipedia	0.211±0.001	0.208±0.000	0.241±0.001	0.235±0.001	0.248±0.002	0.245±0.001	**0.200±0.000**

Table 1 shows that ensembling, concatenation, and FiLM each improve backbone CRPS on four datasets, while the one- and two-block residual variants each improve three. Full-depth residual control is the only mechanism to improve all eight, and it is best or tied-best on five. Concatenation attains the lowest CRPS on Exchange and Solar, and ensembling on UberTLC. Within the residual variants, full-depth control outperforms both shallower designs, except for a tie with the two-block variant on Exchange after rounding. Appendix B.6 provides the complete summary.

Zero-initialized residual projections preserve the pretrained backbone's predictions at the start of training. Appendix B.12 provides further ablations of projection initialization, controller initialization, and controller depth.

Table 2: **Forecasting results on eight benchmarks.** CRPS; lower is better. Best/second-best values are **bold**/underlined, including ties. Our evaluations report mean±SD over three independent runs with different random seeds.

Method	Electricity	Exchange	KDDCup	M4-Hourly	Solar	Traffic	UberTLC	Wikipedia
ARIMA	0.344±0.000	**0.008±0.000**	0.514±0.000	0.031±0.000	0.558±0.003	0.486±0.000	0.478±0.000	0.654±0.000
DLinear	0.058±0.001	0.015±0.004	0.318±0.015	0.055±0.007	0.794±0.027	0.131±0.000	0.250±0.006	0.259±0.002
DeepAR	0.051±0.000	0.013±0.004	0.362±0.017	0.046±0.006	0.429±0.055	0.103±0.002	0.169±0.002	0.215±0.003
TFT	0.060±0.001	0.011±0.000	0.543±0.048	0.038±0.002	0.371±0.006	0.128±0.005	0.202±0.009	0.219±0.004
PatchTST	0.055±0.001	0.010±0.001	0.420±0.011	0.034±0.004	0.728±0.015	0.151±0.007	0.219±0.004	0.209±0.001
iTransformer	0.074±0.000	0.010±0.000	-	-	0.496±0.000	0.158±0.000	-	0.262±0.000
K^2VAE	0.078±0.000	0.014±0.000	0.332±0.000	0.046±0.000	0.371±0.000	0.126±0.000	0.178±0.001	0.291±0.000
CSDI	0.051±0.000	0.013±0.001	0.309±0.006	0.043±0.004	0.360±0.006	0.152±0.001	0.213±0.007	0.318±0.012
SSSD	0.048±0.001	0.010±0.001	0.274±0.009	0.050±0.007	0.384±0.023	0.097±0.002	0.156±0.007	0.209±0.004
Biloš et al.	0.067±0.002	0.012±0.004	1.147±0.300	-	0.379±0.009	0.317±0.053	0.450±0.086	0.318±0.022
CN-Diff	0.084±0.000	0.011±0.000	0.470±0.000	0.050±0.000	0.535±0.002	0.173±0.000	0.471±0.000	0.397±0.000
TSDiff	0.049±0.000	0.012±0.001	0.312±0.001	0.034±0.000	0.402±0.000	0.099±0.000	0.172±0.000	0.211±0.001
TSDiff ML-Q	0.049±0.000	0.011±0.000	0.360±0.002	0.038±0.000	0.399±0.000	0.094±0.000	0.172±0.000	0.215±0.000
TSDiff LMC-Q	0.051±0.000	0.011±0.001	0.361±0.002	0.039±0.000	0.400±0.000	0.097±0.000	0.172±0.001	0.215±0.000
TSFlow-Ucond.	0.049±0.000	0.011±0.001	0.287±0.005	0.032±0.001	0.396±0.005	0.086±0.000	**0.154±0.002**	0.291±0.005
TSFlow-Cond. (OU)	**0.045±0.000**	**0.008±0.000**	0.288±0.004	0.028±0.008	0.344±0.006	**0.082±0.000**	**0.154±0.002**	0.207±0.001
DeepAR-Large	0.047±0.001	0.022±0.006	0.380±0.019	0.047±0.006	0.371±0.020	0.092±0.005	0.164±0.005	0.227±0.006
TFT-Large	0.048±0.000	0.010±0.001	0.354±0.044	0.030±0.003	0.552±0.045	0.099±0.010	0.161±0.002	0.205±0.001
TSDiff-Wide	0.048±0.000	0.012±0.000	0.345±0.002	0.035±0.000	0.409±0.001	0.095±0.000	0.167±0.000	0.222±0.000
Control-TS-Q1	0.049±0.000	**0.008±0.000**	0.282±0.001	0.025±0.000	0.408±0.000	0.098±0.000	0.170±0.000	0.203±0.000
Control-TS-Q3	0.048±0.000	**0.008±0.000**	**0.249±0.001**	0.026±0.000	0.392±0.000	0.097±0.000	0.169±0.000	0.202±0.000
Control-TS-Q5	0.047±0.000	**0.008±0.000**	0.256±0.001	**0.022±0.000**	**0.341±0.000**	0.100±0.000	0.172±0.000	**0.200±0.000**

3.2 Forecasting performance and applicability

Forecasting accuracy. Table 2 compares Control-TS with established forecasters across eight GluonTS benchmarks using CRPS under a shared univariate protocol. Details of our K^2VAE and CN-Diff evaluations are provided in Appendix B.8. Control-TS selects its external forecaster by validation CRPS for each dataset and quantile set, and it improves on TSDiff across all eight benchmarks, with the largest reduction of approximately 35% on M4-Hourly (0.034 to 0.022). It achieves the best or tied-best mean CRPS among all evaluated forecasters on five benchmarks; TSFlow performs better on Electricity, Traffic, and UberTLC. Control-TS also outperforms the external forecaster on most datasets, supporting the value of integrating its predictions with the diffusion backbone.

Training-free refinement. TSDiff's refinement mode combines an external forecaster with the frozen backbone without training: it refines the forecaster's samples under the backbone's learned density (Kollovieh et al., 2023). Initialized with DeepAR samples, its two quantile-regularized variants (TSDiff ML-Q and LMC-Q; Appendix B.9) improve on TSDiff only on Exchange, Solar and Traffic, whereas Control-TS outperforms both on every dataset except Traffic. This suggests that learning how quantiles condition denoising offers benefits beyond post-hoc refinement.

Capacity controls. TSDiff-Wide doubles the backbone width from 64 to 128, giving 498-502k parameters. Complete Control-TS uses 408-506k, including its external forecaster; its diffusion component alone uses 380-385k. Control-TS achieves lower CRPS than TSDiff-Wide on six datasets, while TSDiff-Wide performs better on Traffic and UberTLC. It also matches or outperforms both DeepAR-Large and TFT-Large on six datasets. These comparisons support gains beyond simply increasing parameter count (Appendix B.5).

Qualitative forecasts. In Figure 3, Control-TS continues the oscillatory pattern in the M4-Hourly example, where TSDiff's median flattens, and yields a sharper, better-centered interval in the Electricity example. Appendix B.18 provides further comparisons.

Applicability to additional backbones. We assess applicability beyond TSDiff using CSDI (Tashiro et al., 2021) with DeepAR controls and multivariate MG-TSD (Fan et al., 2024) with DeepVAR controls (Salinas et al., 2019). Each backbone retains its native forecasting computation while incorporating the quantile-driven residual pathway. Quantile control reduces CSDI's CRPS by 0.34%-11.05% across all eight datasets and improves MG-TSD's CRPS-sum on five of eight (Table 3). These results extend the observed benefits beyond TSDiff, with their magnitude and consistency depending on the backbone and dataset.

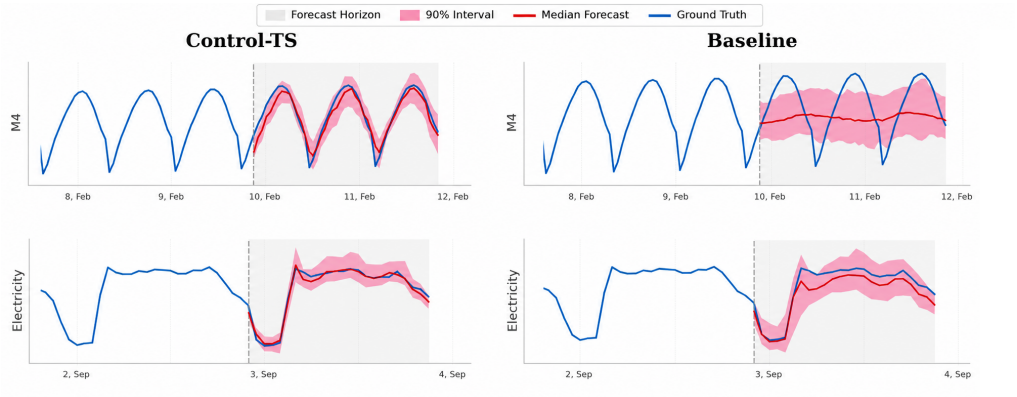


Figure 3: **Qualitative predictive distributions.** Control-TS and TSDiff medians and 90% intervals on shared M4-Hourly and Electricity contexts. Shaded regions denote the forecast horizon.

Table 3: **Quantile control on additional diffusion backbones.** CRPS for univariate CSDI; CRPS-sum for multivariate MG-TSD. Results are based on three independent runs with different random seeds. Positive Δ denotes relative improvement over the corresponding backbone. Scores are not pooled across backbones or with the main benchmark comparison. Appendix B.17 provides detailed results and protocols.

Backbone	Metric	Improved datasets	Range Δ (%)
CSDI	CRPS	8/8	$[+0.34, +11.05]$
MG-TSD	CRPS-sum	5/8	$[-7.23, +38.27]$

3.3 Role of the Control Information

Having examined how controls enter the denoising network, we next investigate what information they provide. We compare predicted quantiles with observed-context controls, then examine quantile granularity and the choice of external forecaster.

Observed context as control. We keep the same full-depth residual architecture and replace the predicted quantiles with the observed context repeated to the same number of channels: once for Q1, three times for Q3, and five times for Q5. This comparison is intended to isolate the quantile forecasts, so that a difference is not attributable to the architecture or to access to the history alone. Across widths, predicted quantiles give lower CRPS in 21 of the 24 dataset–granularity comparisons. The repeated context is lower for Electricity Q1, Solar Q1, and Traffic Q5 (Appendix B.11).

Quantile granularity. We vary the granularity of the explicit forecast through Q1, Q3, and Q5 (Table 2). Validation selects Q3 or Q5 on every dataset, with large gains on KDDCup, M4-Hourly, and Solar. Multi-quantile variants do not uniformly improve over Q1, however, and increasing Q changes both information content and input width. The results therefore support richer quantile representations without isolating the contributions of location, dispersion, and asymmetry.

External forecaster. The complete DeepAR/TFT $\times$ Q1/Q3/Q5 grid in Appendix B.10 shows that the preferred control source depends on the dataset. TFT supplies the selected controls on five benchmarks, while DeepAR supplies them on M4-Hourly, Solar, and Traffic.

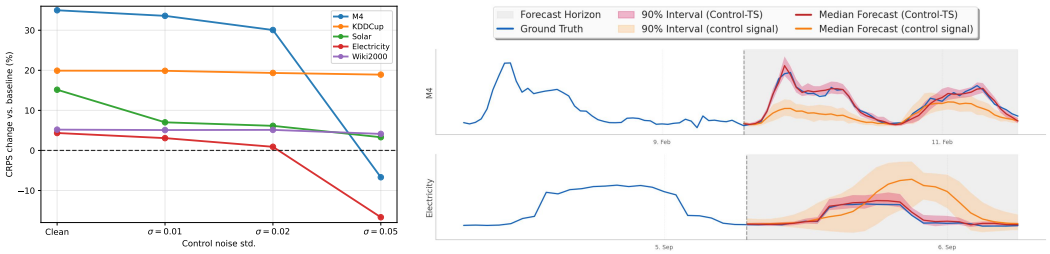
3.4 Robustness to Degraded Controls

Gaussian corruption. We assess sensitivity to control quality by adding Gaussian noise to the control signal at inference in normalized control space, with the controller and backbone fixed. Figure 4a reports the percentage CRPS improvement over the backbone on representative datasets; positive values indicate retained gains. Performance degrades smoothly: gains are preserved under mild to moderate corruption but become less consistent under stronger noise, as expected when the quantile signal becomes weakly informative. Appendix B.13 provides the full results.

Table 4: **Train-on-Hybrid, Test-on-Real CRPS.** Mean±SD over three independent runs with different random seeds. The better synthetic source is **bold**, including ties.

		Electricity	Exchange	KDDCup	M4-Hourly	Solar	Traffic	UberTLC	Wikipedia
Linear (CLPS)	TSDiff	**0.070±0.001**	0.096±0.000	0.900±0.001	0.041±0.000	0.543±0.002	0.157±0.001	**0.303±0.003**	0.392±0.006
	Control-TS	0.072±0.004	**0.028±0.000**	**0.699±0.006**	**0.038±0.001**	**0.526±0.003**	**0.154±0.001**	0.306±0.001	**0.374±0.007**
DeepAR	TSDiff	**0.054±0.002**	**0.011±0.003**	0.480±0.027	**0.025±0.001**	**0.380±0.016**	0.116±0.002	0.174±0.002	0.224±0.006
	Control-TS	**0.054±0.001**	0.012±0.008	**0.411±0.033**	0.035±0.007	0.387±0.009	**0.110±0.002**	**0.173±0.006**	**0.223±0.005**
TFT	TSDiff	**0.058±0.001**	0.015±0.002	0.818±0.099	**0.032±0.002**	0.374±0.007	0.137±0.007	0.238±0.011	0.232±0.003
	Control-TS	0.060±0.001	**0.007±0.000**	**0.632±0.001**	**0.032±0.001**	**0.364±0.007**	**0.124±0.014**	**0.221±0.010**	**0.230±0.001**

Naturally misaligned controls. We also examine cases in which the predicted quantile scaffold is poorly aligned with the realized future. The examples in Figure 4b show that Control-TS does not simply track this unreliable scaffold: its samples remain shaped by the diffusion backbone while using the quantile estimates as guidance. Together, the Gaussian-corruption results and the examples with naturally low-quality controls indicate that Control-TS uses the auxiliary uncertainty as guidance rather than as a hard target. Appendix B.14 includes failure cases in which the forecast follows a misaligned scaffold.



(a) Gaussian control perturbations.

(b) Examples under misaligned control signals.

Figure 4: **Robustness to degraded control signals.** Left: CRPS improvement over TSDiff as Gaussian corruption increases; positive values indicate retained gains. Right: selected M4-Hourly and Electricity examples with inaccurate quantile scaffolds; Control-TS stays closer to the realized continuation than the scaffold. Full corruption results and failure cases are in Appendices B.13, B.14.

3.5 Forecasting Signal in Generated Futures

Under Train-on-Hybrid, Test-on-Real (THTR) (Esteban et al., 2017), a forecaster trains on real contexts paired with generated futures and is evaluated on held-out real data using CRPS. CLPS uses a shared ridge evaluator with inexpensive, deterministic fitting for fixed data to assess the forecasting signal accessible to that evaluator. The *Linear* block of Table 4 reports CLPS; DeepAR and TFT probe nonlinear forecasters using the same training pairs. Appendix B.16 provides the protocol.

Control-TS improves CLPS over TSDiff on six datasets, most strongly on Exchange and KDDCup, with small losses on Electricity and UberTLC. The DeepAR and TFT evaluations show gains on four and six datasets, respectively, each with one additional tie.

3.6 Computational Cost

Forecast generation. We compare full 100-step forecast calls at batch size 16, including online quantile generation. Control-TS increases wall-clock time by 17-61% (30% on average), FLOPs by 7-25% (16.5% on average), and peak VRAM by 2-7%. The interface therefore trades additional computation for improved forecasting, with comparatively modest inference-memory overhead.

Controller training. During controller fitting with cached controls, per-step time overhead is below 15% on six datasets and below 66% on all eight; peak VRAM increases by approximately 65%. These training costs are incremental to the pretrained backbone and external forecaster. Appendix B.15 provides the full accounting.

4 Related work

Explicit probabilistic forecasting. Classical and neural forecasters expose uncertainty through likelihoods, quantiles, or intervals (Makridakis et al., 2020; Smyl, 2020). Examples include ARIMA (Hyndman & Khandakar, 2008), DeepAR (Salinas et al., 2020), TFT (Lim et al., 2021), DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2023), iTransformer (Liu et al., 2024), and global state-space or factorized models (Rangapuram et al., 2018; Wang et al., 2019; Sen et al., 2019). We study predicted quantiles as controls for a separate implicit forecaster, rather than only as final summaries.

Implicit generative forecasting. Diffusion modeling (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021b;a) supports autoregressive forecasting (Rasul et al., 2021), score-based imputation (Tashiro et al., 2021; Lopez Alcaraz & Strodthoff, 2023), and guided trajectory generation (Biloš et al., 2023; Kollovieh et al., 2023); TSFlow uses flow matching (Kollovieh et al., 2025). Recent models include K^2VAE (Wu et al., 2025) and CN-Diff (Rishi et al., 2025). Others condition diffusion on point forecasts: a pretrained mean estimator in CARD (Han et al., 2022), transformer forecasts in TMDM (Li et al., 2024), and linear autoregressive initialization in TimeDiff (Shen & Kwok, 2023). TSDiff refines base forecasts under its learned density without training (Kollovieh et al., 2023). We instead learn to condition a frozen backbone on externally predicted quantile trajectories and compare integration mechanisms, including TSDiff refinement (Section 3.2).

Conditioning and control. Classifier and classifier-free guidance steer sampling (Dhariwal & Nichol, 2021; Ho & Salimans, 2022); FiLM modulates features (Perez et al., 2018). Frozen-backbone adaptation includes parallel residual adapters (Rebuffi et al., 2018) and side-tuning (Zhang et al., 2020). Earlier diffusion implementations also use zero-initialized output layers within residual branches (Nichol & Dhariwal, 2021). For spatial conditioning in image generation, ControlNet connects a frozen denoiser to a trainable copy of its pretrained encoder and middle block through zero-initialized projections (Zhang et al., 2023). Control-TS builds on this auxiliary residual-conditioning principle, while initializing its controller independently of the pretrained backbone. The temporal controller maps horizon-aligned quantile forecasts and the diffusion timestep to residual corrections throughout the frozen denoising network. Our study focuses on predicted quantiles as controls that convey marginal forecast uncertainty, investigating how this information can effectively guide diffusion forecasting through a comparison of alternative integration mechanisms.

Evaluation and downstream forecasting. We use CRPS for forecasting evaluation (Gneiting & Raftery, 2007; Hersbach, 2000). CLPS adapts TSDiff's linear predictive score (Kollovieh et al., 2023) to real-context/synthetic-future pairs in Train-on-Hybrid, Test-on-Real evaluation (Esteban et al., 2017), testing the forecasting signal in generated futures.

5 Conclusion

We established predicted quantile trajectories as explicit distributional controls for implicit diffusion-based trajectory generation, comparing input conditioning, feature modulation, residual control, and output-level ensembling. Full-depth residual control (Control-TS) yielded the most consistent improvements on the primary backbone, motivating ablations and analyses of quantile granularity, predictor choice, controller design, robustness, and computational cost. Control-TS is competitive with strong probabilistic forecasters, reducing backbone CRPS on all eight benchmarks by up to approximately 35% and outperforming a matched-capacity diffusion baseline on six. Experiments with CSDI and MG-TSD support broader applicability, with benefits varying across backbones and datasets. We introduce CLPS, a Train-on-Hybrid, Test-on-Real (THTR) criterion, to complement forecast accuracy by assessing the downstream forecasting utility of generated futures.

6 Limitations and Future Work

Our study focuses on quantile trajectories predicted by learned forecasters. Inaccurate or misaligned controls may still degrade forecasting performance. The use of other predictive summaries (e.g., moments or temporal correlations), domain-expert knowledge, and system constraints as control signals remains unexplored. Future work will investigate how to represent and incorporate these signals, alongside adaptive gating to adjust the strength of quantile guidance at each injection.

AI USE STATEMENT

Generative AI tools were used to assist with drafting and polishing the manuscript, including improving grammar, clarity, concision, and organization, and to assist with the refinement and organization of existing code. The authors reviewed and revised the AI-assisted text and code to ensure that the manuscript accurately reflected the methods, experimental findings, and conclusions. The authors take full responsibility for the final manuscript, including the accuracy of its claims, results, and references.

REFERENCES

ACM SIGKDD. KDD cup 2018: KDD cup of fresh air. https://www.kdd.org/kdd2018/kdd-cup, 2018.

Alexander Alexandrov, Konstantinos Benidis, Michael Bohlke-Schneider, Valentin Flunkert, Jan Gasthaus, Tim Januschowski, Danielle C. Maddix, Syama Sundar Rangapuram, David Salinas, Jasper Schulz, Lorenzo Stella, Ali Caner Turkmen, and Yuyang Wang. Probabilistic and neural time series modeling in python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL https://jmlr.org/papers/v21/19-820.html.

Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling temporal data as continuous functions with stochastic process diffusion. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

C. Bui, N. Pham, Anh Vo, A. Tran, A. Nguyen, and Trung Le. Time series forecasting for healthcare diagnosis and prognostics with the focus on cardiovascular diseases. In *6th International Conference on the Development of Biomedical Engineering in Vietnam (BME6)*, volume 63 of *IFMBE Proceedings*, pp. 809–818. Springer, 06 2018. ISBN 978-981-10-4360-4. doi: 10.1007/978-981-10-4361-1_138.

Prafulla Dhariwal and Alexander Quinn Nichol. Diffusion models beat gans on image synthesis. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021 (NeurIPS 2021)*, pp. 8780–8794, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/49ad23d1ec9fa4bd8d77d02681df5cfa-Abstract.html.

Dheeru Dua and Casey Graff. UCI machine learning repository. https://archive.ics.uci.edu/, 2017.

Cristóbal Esteban, Stephanie Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017. doi: 10.48550/arXiv.1706.02633. URL https://arxiv.org/abs/1706.02633.

Xinyao Fan, Yueying Wu, Chang Xu, Yuhao Huang, Weiqing Liu, and Jiang Bian. MG-TSD: Multi-granularity time series diffusion models with guided learning process. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=CZiY6OLktd.

Robert Fildes, Shaohui Ma, and Stephan Kolassa. Retail forecasting: Research and practice. *International Journal of Forecasting*, 38(4):1283–1318, 2022. doi: 10.1016/j.ijforecast.2019.06.004.

FiveThirtyEight. Uber pickups in new york city. https://www.kaggle.com/datasets/fivethirtyeight/uber-pickups-in-new-york-city, 2016.

Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. Probabilistic forecasting with spline quantile function RNNs. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, volume 89 of *Proceedings of Machine Learning Research*, pp. 1901–1910. PMLR, 2019. URL https://proceedings.mlr.press/v89/gasthaus19a.html.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437.

Rakshitha Godahewa, Christoph Bergmeir, Geoff Webb, Rob Hyndman, and Pablo Montero-Manso. Kdd cup dataset (without missing values). https://doi.org/10.5281/zenodo.4656756, 2020.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/pdf?id=uYLFoz1vlAC.

Xizewen Han, Huangjie Zheng, and Mingyuan Zhou. CARD: Classification and regression diffusion models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 18100–18115, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/72dad95a24fae750f8ab1cb3dab5e58d-Abstract-Conference.html.

Hans Hersbach. Decomposition of the continuous ranked probability score for ensemble prediction systems. *Weather and Forecasting*, 15(5):559–570, 2000.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Rob J. Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, 3 edition, 2021. URL https://otexts.com/fpp3/.

Rob J. Hyndman and Yeasmin Khandakar. Automatic time series forecasting: The forecast package for r. *Journal of Statistical Software*, 27(3):1–22, 2008. doi: 10.18637/jss.v027.i03.

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

Marcel Kollovieh, Abdul Fatir Ansari, Michael Böhlke-Schneider, Jasper Zschiegner, Hao Wang, and Yuyang Wang. Predict, refine, synthesize: Self-guiding diffusion models for probabilistic time series forecasting. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.neurips.cc/paper_files/paper/2023/file/5a1a10c2c2c9b9af1514687bc24b8f3d-Paper-Conference.pdf.

Marcel Kollovieh, Marten Lienen, David Lüdke, Leo Schwinn, and Stephan Günnemann. Flow matching with gaussian process priors for probabilistic time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=uxVBbSlKQ4.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *International Conference on Learning Representations (ICLR)*, 2021.

Guokun Lai, Wei-Cheng Chang, Yiming Yang, and Hanxiao Liu. Modeling long- and short-term temporal patterns with deep neural networks. In *Proceedings of the 41st International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 95–104. ACM, 2018.

Yuxin Li, Wenchao Chen, Xinyue Hu, Bo Chen, Baolin Sun, and Mingyuan Zhou. Transformer-modulated diffusion models for probabilistic multivariate time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=qae04YACHs.

Bryan Lim, Sercan Ö. Arik, Nicolas Loeff, and Tomas Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. doi: 10.1016/j.ijforecast.2021.03.012.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Juan Miguel Lopez Alcaraz and Nils Strodthoff. Diffusion-based time series imputation and forecasting with structured state space models. *Transactions on Machine Learning Research*, 2023. URL https://openreview.net/forum?id=hHiIbk7ApW.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1): 54–74, 2020. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.04.014. URL https://doi.org/10.1016/j.ijforecast.2019.04.014. M4 Competition.

Manfred Mudelsee. Trend analysis of climate time series: A review of methods. *Earth-Science Reviews*, 190:310–322, 2019. ISSN 0012-8252. doi: 10.1016/j.earscirev.2018.12.005. URL https://doi.org/10.1016/j.earscirev.2018.12.005.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning (ICML 2021)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8162–8171. PMLR, 2021. URL http://proceedings.mlr.press/v139/nichol21a.html.

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

Art B. Owen. *Monte Carlo theory, methods and examples*. https://artowen.su.domains/mc/, 2013.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. Film: visual reasoning with a general conditioning layer. AAAI'18/IAAI'18/EAAI'18. AAAI Press, 2018. ISBN 978-1-57735-800-8.

Syama Sundar Rangapuram, Matthias W. Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *Advances in Neural Information Processing Systems*, volume 31, pp. 7796–7805, 2018. URL https://papers.nips.cc/paper/8004-deep-state-space-models-for-time-series-forecasting.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8857–8868. PMLR, 2021.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8119–8127, 2018. doi: 10.1109/CVPR.2018.00847.

J Rishi, GVS Mothish, and Deepak Subramani. Conditional diffusion model with nonlinear data transformation for time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=kcUNMKqrCg.

David Salinas, Michael Böhlke-Schneider, Laurent Callot, Roberto Medico, and Jan Gasthaus. High-dimensional multivariate forecasting with low-rank gaussian copula processes. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://arxiv.org/abs/1910.03002.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3): 1181–1191, 2020. doi: 10.1016/j.ijforecast.2019.07.001.

Rajat Sen, Hsiang-Fu Yu, and Inderjit S. Dhillon. Think globally, act locally: A deep neural network approach to high-dimensional time series forecasting. In *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 4838–4847, 2019.

Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning : A systematic literature review: 2005–2019. *Applied Soft Computing*, 90:106181, 2020. ISSN 1568-4946. doi: 10.1016/j.asoc.2020.106181. URL https://doi.org/10.1016/j.asoc.2020.106181.

Lifeng Shen and James Kwok. Non-autoregressive conditional diffusion models for time series prediction. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 31016–31029. PMLR, 2023. URL https://proceedings.mlr.press/v202/shen23d.html.

Slawek Smyl. A hybrid method of exponential smoothing and recurrent neural networks for time series forecasting. *International Journal of Forecasting*, 36(1):75–85, 2020. doi: 10.1016/j.ijforecast.2019.03.017.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, volume 37 of *Proceedings of Machine Learning Research*, pp. 2256–2265. PMLR, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations (ICLR)*, 2021a.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021b.

Yusuke Tashiro, Jiaming Song, Yang Song, and Stefano Ermon. CSDI: conditional score-based diffusion models for probabilistic time series imputation. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021 (NeurIPS 2021)*, pp. 24804–24816, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/cfe8504bda37b575c70ee1a8276f3486-Abstract.html.

Yuyang Wang, Alex Smola, Danielle C. Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep factors for forecasting. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6607–6617. PMLR, 2019.

Xingjian Wu, Xiangfei Qiu, Hongfan Gao, Jilin Hu, Bin Yang, and Chenjuan Guo. K^2VAE: A koopman-kalman enhanced variational autoencoder for probabilistic time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2505.23017.

Changtian Ying, Weiqing Wang, Jiong Yu, Qi Li, Donghua Yu, and Jianhua Liu. Deep learning for renewable energy forecasting: A taxonomy, and systematic literature review. *Journal of Cleaner Production*, 384:135414, 01 2023. doi: 10.1016/j.jclepro.2022.135414.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, 2023. doi: 10.1609/aaai.v37i9.26317.

Jeffrey O. Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. Side-tuning: A baseline for network adaptation via additive side networks. In *Computer Vision – ECCV 2020*, Lecture Notes in Computer Science, pp. 698–714. Springer, 2020. doi: 10.1007/978-3-030-58580-8_41.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models. 10 2023. doi: 10.1109/ICCV51070.2023.00355.

A APPENDICES

A.1 DIFFUSION EQUATIONS, SCHEDULES, AND OBSERVATION SELF-GUIDANCE

We provide the complete DDPM formulation (Ho et al., 2020), including the forward and reverse diffusion processes, schedule definitions, the training objective, and the explicit formulation of **observation self-guidance** used during inference. Observation self-guidance was introduced in TSDiff (Kollovieh et al., 2023); we restate it here for completeness and to fix notation in the controlled setting.

Forward process and schedule. TSDiff follows a standard forward diffusion process

$$q(x_t \mid x_0) = \mathcal{N}\big(\sqrt{\bar{\alpha}_t}\, x_0,\ (1-\bar{\alpha}_t)I\big), \tag{4}$$

$$x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\, \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{5}$$

where the schedule is defined by $\beta_t \in (0,1)$, $\alpha_t = 1-\beta_t$, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$.

DDPM training objective. The unconditional backbone is trained using the standard DDPM objective

$$\mathcal{L}_{\text{diff}}(\theta_B) = \mathbb{E}_{x_0,\, t,\, \epsilon}\Big[\big\|\epsilon - \epsilon_{\theta_B}(x_t, t)\big\|_2^2\Big], \tag{6}$$

$$\text{where } t \sim \text{Unif}\{1, \ldots, T_{\text{diff}}\},\ \epsilon \sim \mathcal{N}(0, I).$$

Reverse process. Reverse diffusion follows the standard DDPM update. Let $\epsilon_\theta(x_t, t)$ be the noise predictor and define the denoised estimate

$$\tilde{x}_0(x_t, t) \ = \ \frac{x_t - \sqrt{1-\bar{\alpha}_t}\, \epsilon_\theta(x_t, t)}{\sqrt{\bar{\alpha}_t}}. \tag{7}$$

We form the Gaussian reverse kernel mean

$$\mu_\theta(x_t, t) \ = \ \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\, \epsilon_\theta(x_t, t)\Big), \tag{8}$$

and sample

$$x_{t-1} \ = \ \mu_\theta(x_t, t) + \sqrt{\tilde{\beta}_t}\, z, \qquad z \sim \mathcal{N}(0, I), \tag{9}$$

where $\tilde{\beta}_t = \beta_t \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$ is the standard posterior variance. For the backbone, these expressions use $\epsilon_{\theta_B}(x_t, t)$; for Control-TS, they use the controlled predictor $\hat{\epsilon}_\theta(x_t, t \mid c)$ with $\theta = (\theta_B, \theta_C)$.

Observation self-guidance. Observation self-guidance enforces consistency with the observed context by modifying the reverse-kernel mean using a guidance score (Kollovieh et al., 2023). Let $m \in \{0,1\}^{L \times D}$ be an observation mask with $m_{i,d} = 1$ for context time indices $i \in \{1, \ldots, L_{\text{ctx}}\}$ and $m_{i,d} = 0$ for forecast time indices $i \in \{L_{\text{ctx}}+1, \ldots, L_{\text{ctx}}+L_{\text{tar}}\}$ (when lag features are enabled, $D > 1$ includes the corresponding lag channels). In Control-TS, observation self-guidance is evaluated through the unmodified frozen backbone, with the Controller disabled. Thus, the denoised estimate used in the guidance energy is the backbone-only estimate $\tilde{x}_{0,B}(x_t, t)$. Given an observation tensor y (context + zero-padded future), the energy is

$$E(x_t, t; y, m) \ = \ \sum_{(i,d):\, m_{i,d}=1} \ell\Big(\tilde{x}_{0,B}(x_t, t)_{i,d},\ y_{i,d}\Big). \tag{10}$$

where ℓ is either mean-squared error (MSE guidance) or the quantile loss used by our sampler (quantile guidance) (Koenker & Bassett, 1978). The corresponding guidance score is the negative energy gradient,

$$g(x_t, t) \ = \ -\nabla_{x_t} E(x_t, t; y, m), \tag{11}$$

which induces a guided reverse kernel of the form

$$p_{\theta_B, \theta_C}(x_{t-1} \mid x_t, y, m, c) \ = \ \mathcal{N}\Big(\mu_{\theta_B, \theta_C}(x_t, t, c) + \lambda_t\, g(x_t, t),\ \tilde{\beta}_t I\Big), \tag{12}$$

with $\lambda_t = s\, \tilde{\beta}_t$ for a user-chosen guidance scale s, matching observation self-guidance in Kollovieh et al. (2023). Here, μ_{θ_B, θ_C} is computed using the controlled denoising prediction, whereas $g(x_t, t)$ is computed from the backbone-only denoised estimate $\tilde{x}_{0,B}(x_t, t)$ in Eq. 10.

A.2 Extended Control-TS Formalization and Architectural Details

This subsection expands the main-text formalization of Control-TS and provides additional architectural details referenced in Section 2.3.

Backbone and Controller streams. Let $\{B^{(k)}_{\theta_B}\}_{k=1}^{K}$ denote the residual blocks of the frozen diffusion backbone, $\{C^{(k)}_{\theta_C}\}_{k=1}^{K}$ the corresponding Controller residual blocks, and $\{Z^{(k)}_{\theta_C}\}_{k=1}^{K}$ the zero-initialized channel projection operators associated with each residual depth. We define the backbone hidden stream $\{h^{(k)}_t\}_{k=0}^{K}$ and the Controller stream $\{u^{(k)}_t\}_{k=0}^{K}$ at diffusion step t, where $h^{(k)}_t$ (resp. $u^{(k)}_t$) denotes the representation after the k-th backbone (resp. Controller) residual block.

Controller initialization. Controller residual blocks are copied during model construction and randomly initialized.

Time embeddings and input projections. We use a shared timestep embedding $\tau_{\theta_B}(t)$ that is injected into every residual block in both branches. Separately, $E_{\theta_B}(\cdot)$ denotes the backbone input projection applied to the currently noised trajectory x_t, and $E_{\theta_C}(\cdot)$ denotes the Controller input projection applied to the control signal c. The two streams are initialized as

$$h^{(0)}_t = E_{\theta_B}(x_t), \qquad u^{(0)}_t = E_{\theta_C}(c). \tag{13}$$

Depth-aligned residual injection. For $k = 1, \dots, K$, the Controller and backbone evolve as

$$u^{(k)}_t = C^{(k)}_{\theta_C}\big(u^{(k-1)}_t,\, \tau_{\theta_B}(t)\big), \tag{14}$$

$$h^{(k)}_t = B^{(k)}_{\theta_B}\Big(h^{(k-1)}_t + Z^{(k)}_{\theta_C}\big(u^{(k)}_t\big),\, \tau_{\theta_B}(t)\Big). \tag{15}$$

The control signal enters only through $u^{(0)}_t = E_{\theta_C}(c)$, after which the Controller produces a depth-aligned sequence of residual states $\{u^{(k)}_t\}$ whose projected residuals are injected additively into the corresponding backbone block $B^{(k)}_{\theta_B}$.

Noise prediction head. Each backbone residual block emits a local latent contribution $s^{(k)}_t$, which is aggregated and mapped to the final noise prediction via the backbone readout head (the final convolutional head in Fig. 2). Writing $s_t = \sum_{k=1}^{K} s^{(k)}_t$, the controlled predictor is

$$\hat{\epsilon}_\theta(x_t, t \mid c) = \mathrm{OutputHead}_{\theta_B}(s_t). \tag{16}$$

Exact reduction to the frozen backbone when control is "off". Define the backbone-only predictor as the controlled model evaluated with all zero-initialized channel projections set to zero:

$$\epsilon_{\theta_B}(x_t, t) \;:=\; \hat{\epsilon}_\theta(x_t, t \mid c)\Big|_{Z^{(1)}_{\theta_C} = \cdots = Z^{(K)}_{\theta_C} \equiv 0}. \tag{17}$$

Because each $Z^{(k)}_{\theta_C}$ is initialized to output zero, Control-TS satisfies

$$\hat{\epsilon}_\theta(x_t, t \mid c) = \epsilon_{\theta_B}(x_t, t) \qquad \text{at initialization,} \tag{18}$$

i.e., it reduces exactly to the pretrained diffusion backbone before learning any control influence. A zero-initialized pointwise output projection likewise appears in the official implementation of Nichol & Dhariwal (2021) (`proj_out` in https://github.com/openai/improved-diffusion/blob/main/improved_diffusion/unet.py).

Interpretation of the control mechanism. For interpretation, let $\delta^{(k)}_t := Z^{(k)}_{\theta_C}(u^{(k)}_t)$ denote the control perturbation injected at depth k. Since the backbone parameters are frozen, the Controller learns to inject these $\delta^{(k)}_t$ vectors to steer the backbone hidden-state trajectory. Although the Controller and backbone are coupled and evaluated in a single forward pass at each diffusion step, it is useful to interpret the resulting aggregated latent representation s_t as the sum of a baseline component and a control-induced deviation:

$$\hat{\epsilon}_\theta(x_t, t \mid c) = \mathrm{OutputHead}_{\theta_B}\Bigg(\underbrace{s_t(\emptyset)}_{\text{Baseline state}} + \underbrace{\sum_{k=1}^{K} \Delta s^{(k)}_t(\delta^{(1:k)}_t)}_{\text{Steering effect}}\Bigg). \tag{19}$$

Here, $s_t(\emptyset)$ is the aggregated latent representation the pretrained backbone would produce without control, and $\Delta s_t^{(k)}$ captures the generally non-linear deviation at depth k induced by the cumulative injections. Equation (19) is intended as an interpretation of the control mechanism, not as an alternative computational graph.

A.3 External Forecasters and Quantile Generation

The control signal is produced by an external probabilistic forecaster implemented in GluonTS. We consider two families: **DeepAR** and the **Temporal Fusion Transformer (TFT)**. Both are trained per dataset to produce probabilistic forecasts over the target horizon of length L_{tar} from the observed context.

Quantiles and variants. At inference time, we extract a set of quantile trajectories over the horizon of length L_{tar} and use them as channels in the control signal. The quantile set is configurable; in our experiments we use Q1 $\{0.5\}$, Q3 $\{0.1, 0.5, 0.9\}$, or Q5 $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ depending on the dataset and ablation setting.

Quantile extraction. Given a trained external forecaster, we draw a moderate number of forecast samples and compute the desired quantiles from the resulting predictive distribution. Concretely, for each requested level q we obtain the trajectory in $\mathbb{R}^{L_{\text{tar}}}$ by evaluating the predictor's quantile function $f \mapsto f.\text{quantile}(q)$.

A.4 Preprocessing, Alignment, and Lag Indices

Scaling. We use per-series mean-absolute scaling with a small floor for numerical stability. In the diffusion model, scaling is applied internally to the context before concatenating context and horizon for diffusion; the same scale is used to normalize both the noising target and any auxiliary quantities. For external forecasters, we similarly normalize each series by its mean absolute value computed from the observed context portion before fitting the probabilistic forecaster.

Control alignment. Let Q be the number of control quantiles. The external forecaster outputs horizon quantiles in scaled space in $\mathbb{R}^{L_{\text{tar}} \times Q}$; we form the aligned control signal $c \in \mathbb{R}^{L \times Q}$ with $L = L_{\text{ctx}} + L_{\text{tar}}$ by replicating the scaled context across Q channels and appending the horizon quantiles.

Lag augmentation. When lag features are enabled for some datasets, we use the default GluonTS lag values for the corresponding data frequency (Alexandrov et al., 2020) to incorporate historical information beyond the modeled window of length $L = L_{\text{ctx}} + L_{\text{tar}}$. We expand both the diffusion input and the control signal with the same lag indices to preserve temporal alignment; the resulting lag features contribute $C-1$ additional channels. For hourly data, where C is the total number of channels after lag augmentation, we use lags $\{24, 48, 72, 96, 120, 144, 168, 336, 504, 672\}$; for daily/business-day data, we use $\{30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330, 360\}$. For multi-quantile control, lag expansion is applied per quantile channel and then concatenated across quantiles.

A.5 Denoiser Loss Reduction Under Control

We plot the denoiser loss on Wikipedia over 1000 training epochs in Fig. 5. Training starts from the exact pretrained TSDiff backbone (control injections are zero at initialization) and only improves as control is introduced; the injected uncertainty-aware scaffold makes the remaining denoising residual easier to learn, often yielding a rapid early drop (visible after the first few epochs). We observe a similar pattern across the other datasets.

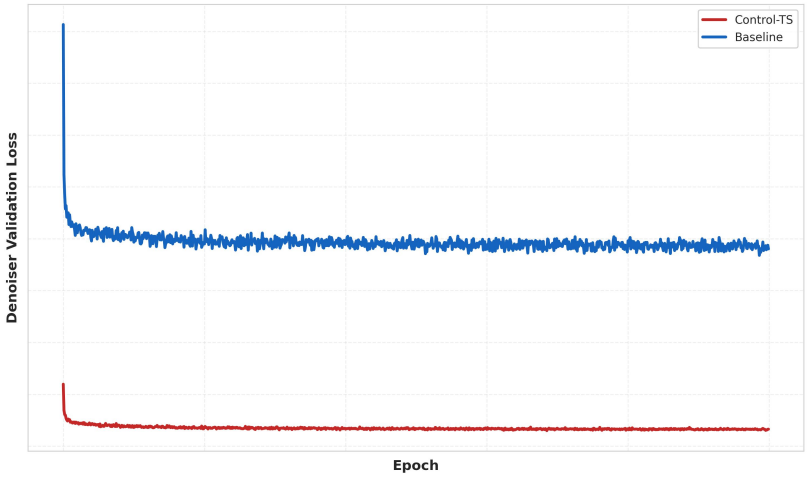


Figure 5: Wikipedia denoiser L2 loss over 1000 epochs.

A.6 Validation CRPS Curves Across Training

We plot validation CRPS during the 1000-epoch Controller training for two representative datasets in Fig. 6, evaluated every 50 epochs. Unlike full test-set evaluation (Table 7), these periodic validation evaluations use fewer Monte Carlo forecast samples for efficiency. We observe a similar pattern across the other datasets.

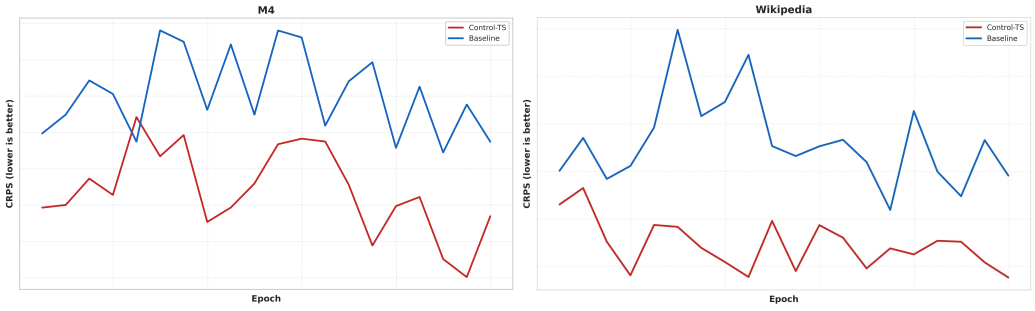


Figure 6: Validation CRPS evaluated every 50 epochs during 1000-epoch Controller training (two example datasets).

A.7 Algorithms (Training and Sampling Pseudocode)

We provide pseudocode for Control-TS training and inference. The training procedure is summarized in Algorithm 1, and inference via controlled reverse diffusion is summarized in Algorithm 2.

Algorithm 1 Control-TS training.

Input: training set; T_{diff}; quantile set $\mathcal{Q}$; pretrained diffusion backbone θ_B; pretrained external-forecaster.
Freeze θ_B.
Initialize controller blocks using the construction-time initialization described in Appendix A.2; initialize all injection-projection weights and biases to zero.
for each optimization step **do**
  Sample batch $(x_0, y_{1:L_{\text{ctx}}})$.
  Sample $t \sim \text{Unif}\{1, \dots, T_{\text{diff}}\}$ and $\epsilon \sim \mathcal{N}(0, I)$.
  $x_t \leftarrow \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \epsilon$.
  $c \leftarrow \text{Align}(\text{Quantiles}(\text{external-forecaster}(y_{1:L_{\text{ctx}}}), \mathcal{Q})) \in \mathbb{R}^{L \times Q}$, $L = L_{\text{ctx}} + L_{\text{tar}}$.
  $\hat{\epsilon} \leftarrow \epsilon_{\theta_B, \theta_C}(x_t, t \mid c)$.
  $\mathcal{L}_{\text{denoise}} \leftarrow \|\epsilon - \hat{\epsilon}\|_2^2$.
  Update θ_C by one Adam step on $\nabla_{\theta_C} \mathcal{L}_{\text{denoise}}$, holding θ_B fixed.
end for

Algorithm 2 Control-TS inference.

Input: context $y_{1:L_{\text{ctx}}}$; quantile set $\mathcal{Q}$; pretrained external-forecaster; the frozen diffusion backbone θ_B and learned θ_C; mask m; N.
$c \leftarrow \text{Align}(\text{Quantiles}(\text{external-forecaster}(y_{1:L_{\text{ctx}}}), \mathcal{Q}))$.
$y \leftarrow [y_{1:L_{\text{ctx}}};\, 0_{1:L_{\text{tar}}}]$ (context + zero-padded future; Appendix A.1).
for $n = 1, 2, \dots, N$ **do**
$x_{T_{\text{diff}}}^{(n)} \sim \mathcal{N}(0, I)$ on $\mathbb{R}^{L \times D}$ with $L = L_{\text{ctx}} + L_{\text{tar}}$.
for $t = T_{\text{diff}}, T_{\text{diff}} - 1, \dots, 1$ **do**
  $\hat{\epsilon} \leftarrow \hat{\epsilon}(x_t^{(n)}, t \mid c)$.
  $\mu_\theta \leftarrow \mu_\theta(x_t^{(n)}, t; \hat{\epsilon})$ (Eq. 8).
  $g \leftarrow g(x_t^{(n)}, t; y, m)$ (Eq. 11).
  $\tilde{\mu}_\theta \leftarrow \mu_\theta + \lambda_t g$ (Eq. 12, $\lambda_t = s\, \tilde{\beta}_t$).
  $z \sim \mathcal{N}(0, I)$ if $t > 1$ else $z \leftarrow 0$.
  $x_{t-1}^{(n)} \leftarrow \tilde{\mu}_\theta + \sqrt{\tilde{\beta}_t}\, z$.
end for
$\hat{\mathbf{y}}_{\text{tar}}^{(n)} \leftarrow$ target-horizon segment of $x_0^{(n)}$.
end for
Return: $\{\hat{\mathbf{y}}_{\text{tar}}^{(n)}\}_{n=1}^{N}$.

Pretrained modules. The diffusion backbone θ_B (frozen throughout Controller training) and the external forecaster are inputs to both procedures; in practice they are taken from off-the-shelf checkpoints in GluonTS (Alexandrov et al., 2020) or other libraries, or trained beforehand and loaded from disk. Only θ_C is optimized in Algorithm 1.

Notation. $\hat{\epsilon}(x_t, t \mid c)$ is the controlled denoiser (frozen backbone θ_B plus learned controller θ_C) at diffusion step t with control c.

B Experimental Details

B.1 Dataset Overview

We use eight commonly used datasets and train/test splits from GluonTS (Alexandrov et al., 2020) spanning multiple domains and sampling frequencies. Table 5 summarizes the dataset statistics.

Table 5: Overview of the datasets and statistics used in our experiments.

Dataset	Train	Test	Domain	Freq.	Median Seq. Len.	Pred. Len.
Electricity[a]	370	2590	$\mathbb{R}^+$	H	5833	24
Exchange[b]	8	40	$\mathbb{R}^+$	D	6071	30
KDDCup[c]	270	270	N	H	10850	48
M4-Hourly[d]	414	414	N	H	960	48
Solar[e]	137	959	$\mathbb{R}^+$	H	7009	24
Traffic[f]	963	6741	$(0, 1)$	H	4001	24
UberTLC[g]	262	262	N	H	4320	24
Wikipedia[h]	2000	10000	N	D	792	30

[a] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[b] https://github.com/laiguokun/multivariate-time-series-data
[c] https://zenodo.org/record/4656756
[d] https://github.com/Mcompetitions/M4-methods/tree/master/Dataset
[e] https://www.nrel.gov/grid/solar-power-data.html
[f] https://zenodo.org/record/4656132
[g] https://github.com/fivethirtyeight/uber-tlc-foil-response
[h] https://github.com/mbohlkeschneider/gluon-ts/tree/mv_release/datasets

B.2 Existing Dataset Assets and Licenses

We used GluonTS only to access preprocessed versions of public benchmark datasets. GluonTS is released under the Apache License 2.0 and documents that its real external datasets are included only when they have a blessed license. Table 6 lists the dataset sources, licenses or available terms of use, and our usage. We do not redistribute raw datasets or third-party dataset archives; all datasets should be obtained from GluonTS or the original sources under their corresponding licenses and terms.

Table 6: Existing dataset assets used in this work. Licenses and terms are reported according to the corresponding official dataset pages, repositories, or documentation pages accessed on 2026-05-07.

Dataset	GluonTS name	Source / provider	License or terms of use
Solar	`solar_nips`	NREL / Monash Time Series Forecasting Repository	CC BY 4.0 via Monash/Zenodo.
Electricity	`electricity_nips`	UCI Machine Learning Repository	CC BY 4.0.
Traffic	`traffic_nips`	Caltrans PeMS / Monash Time Series Forecasting Repository	CC BY 4.0 via Monash/Zenodo.
Exchange	`exchange_rate_nips`	LSTNet multivariate time-series repository	Research use stated by the source repository; no formal named license identified.
M4-Hourly	`m4_hourly`	M4 Competition / Monash Time Series Forecasting Repository	CC BY 4.0 via Monash/Zenodo.
KDDCup	`kdd_cup_2018_without_missing`	KDD Cup 2018 / Monash Time Series Forecasting Repository	CC BY 4.0 via Monash/Zenodo.
UberTLC	`uber_tlc_hourly`	FiveThirtyEight / NYC Taxi & Limousine Commission	CC BY 4.0 unless otherwise noted by FiveThirtyEight.
Wikipedia	`wiki2000_nips`	Wikimedia analytics datasets	CC0 unless otherwise specified by Wikimedia.
GluonTS	Dataset access utility	Amazon / GluonTS	Apache License 2.0.

B.3 CRPS and Sampling Details

CRPS (Gneiting & Raftery, 2007; Hersbach, 2000) is the primary proper scoring rule used for probabilistic time-series forecasting (reported as a loss; lower is better). Intuitively, CRPS measures

how well the *full marginal predictive distribution* at each future time step matches the realized value: it rewards forecasts that are both *accurate* (well-centered) and *well-calibrated* (appropriate dispersion), and it penalizes distributions that are systematically biased or mis-scaled.

For a predictive CDF F and realized value y, CRPS is

$$\mathrm{CRPS}(F, y) = \int_{-\infty}^{\infty} \big(F(z) - \mathbb{I}\{z \geq y\}\big)^2 \, dz. \tag{20}$$

An equivalent algebraic form is

$$\mathrm{CRPS}(F, y) \;=\; \mathbb{E}|X - y| \;-\; \tfrac{1}{2}\mathbb{E}|X - X'|, \qquad X, X' \sim F \text{ i.i.d.} \tag{21}$$

Given Monte Carlo forecast samples $\{\hat{y}^{(i)}\}_{i=1}^{N}$ from F, a standard sample-based estimator is

$$\widehat{\mathrm{CRPS}}(y; \{\hat{y}^{(i)}\}_{i=1}^{N}) = \frac{1}{N}\sum_{i=1}^{N} \big|\hat{y}^{(i)} - y\big| \;-\; \frac{1}{2N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} \big|\hat{y}^{(i)} - \hat{y}^{(j)}\big|. \tag{22}$$

We use CRPS consistently for forecasting evaluation and CLPS evaluation.

For full test-set evaluation, we form the predictive distribution from $N = 100$ Monte Carlo forecast samples per context; during training, periodic validation uses $N = 16$ (and $N = 4$ for Traffic/Wikipedia), following Table 7. The diffusion process uses $T = 100$ steps with a linear β schedule (from 10^{-4} to 10^{-1}).

B.4 Control-TS Hyperparameters

Table 7: Summary of key hyperparameters used by our implementation and configs.

Component	Setting
Hardware	NVIDIA GeForce RTX 2080 Ti
Diffusion steps	$T_{\text{diff}} = 100$
β schedule	linear from 10^{-4} to 10^{-1}
Backbone block	S4 residual blocks (bidirectional), $d_{\text{state}} = 128$
Residual depth	$K = 3$ (small configuration)
Hidden width	64
Time embedding dim	128
Optimizer	Adam
Learning rate	10^{-3}
Gradient clipping	0.5
Training budget	up to 1000 epochs; periodic validation CRPS every 50 epochs
Batching	128 batches per epoch; batch size 64
Checkpoint selection	validation CRPS
Context length	312 (KDDCup/M4-Hourly), 336 (Electricity/Solar/Traffic/UberTLC), 360 (Exchange/Wikipedia)
Guidance type	quantile guidance (default)
Guidance scale	dataset-dependent (e.g., 2)
Control quantiles	Q1/Q3/Q5 depending on experiment
Quantile extraction	100 paths per context for DeepAR; direct quantiles for TFT
CRPS samples (test/val)	$N = 100$ / $N = 16$ ($N = 4$ for Traffic/Wikipedia validation)

Dataset-specific architecture. For Traffic, we use an additional direct x_t connection at the output head.

B.5 Capacity-Matched Diffusion Baseline

TSDiff-Wide increases hidden width from 64 to 128, yielding 498-502k parameters versus 408-506k for Control-TS, including the external forecaster, which accounts for approximately 7-24% of the total depending on the configuration. It uses the same data protocol, validation-CRPS checkpoint

selection, and 100-path evaluation. We report mean and standard deviation over three independent runs with different random seeds. Table 8 gives the complete-system comparison.

Table 8: Complete-system capacity comparison: mean±SD. Control-TS includes the backbone, Controller, and external forecaster; its results are from Table 2. Best mean CRPS is **bold**.

	TSDiff-Wide		Control-TS	
Dataset	Params [k]	CRPS	Params [k]	CRPS
Electricity	501.4	0.048±0.000	505.6	**0.047±0.000**
Exchange	501.5	0.012±0.000	493.1	**0.008±0.000**
KDDCup	501.4	0.345±0.002	504.2	**0.249±0.001**
M4-Hourly	498.8	0.035±0.000	407.9	**0.022±0.000**
Solar	501.4	0.409±0.001	412.4	**0.341±0.000**
Traffic	501.4	**0.095±0.000**	411.0	0.097±0.000
UberTLC	501.4	**0.167±0.000**	504.2	0.169±0.000
Wikipedia	498.4	0.222±0.000	489.4	**0.200±0.000**

DeepAR-Large and TFT-Large match the complete-system budget; Table 9 reports three independent training runs, each evaluated once.

Table 9: Capacity-comparable explicit forecasters: mean±SD. Control-TS results are from Table 2. Best values are **bold**.

	DeepAR-Large		TFT-Large		Control-TS	
Dataset	Params [k]	CRPS	Params [k]	CRPS	Params [k]	CRPS
Electricity	502.7	**0.047±0.001**	493.5	0.048±0.000	505.6	**0.047±0.000**
Exchange	496.5	0.022±0.006	500.3	0.010±0.001	493.1	**0.008±0.000**
KDDCup	502.7	0.380±0.019	493.5	0.354±0.044	504.2	**0.249±0.001**
M4-Hourly	502.7	0.047±0.006	493.5	0.030±0.003	407.9	**0.022±0.000**
Solar	502.7	0.371±0.020	493.5	0.552±0.045	412.4	**0.341±0.000**
Traffic	502.7	**0.092±0.005**	493.5	0.099±0.010	411.0	0.097±0.000
UberTLC	502.7	0.164±0.005	493.5	**0.161±0.002**	504.2	0.169±0.000
Wikipedia	497.0	0.227±0.006	500.3	0.205±0.001	489.4	**0.200±0.000**

B.6 Interaction-Study Summary

Table 10 counts datasets on which each mechanism has strictly lower CRPS than the TSDiff backbone in Table 1.

Table 10: Datasets on which each interaction mechanism improves the backbone.

Mechanism	Better than the backbone
Ensembling	4/8
Input concatenation	4/8
FiLM	4/8
1-block residual	3/8
2-block residual	3/8
Control-TS	**8/8**

B.7 Frozen and Fine-Tuned Conditioning

Table 11 separates fine-tuned and frozen-backbone variants of input concatenation and FiLM. The fine-tuned entries are those reported in Table 1.

As visible in the results, holding the backbone fixed is a less favorable setting for concatenation and FiLM. Residual control, including the one-block and two-block variants as well as full-depth Control-TS, trains a separate pathway and keeps the backbone fixed. Input concatenation and FiLM have no such pathway. Concatenation appends the quantile trajectories to the noisy denoiser input, so they are read by the backbone's input projection. FiLM maps them to scale and shift applied to intermediate backbone features. In both cases the quantile interface is carried by backbone parameters. Freezing those parameters removes the weights that would adapt to the conditioning, which is consistent with the weaker frozen-backbone results. The comparison reported in the main text therefore uses the fine-tuned denoiser. Even with the backbone held fixed, some runs improve on the backbone's results.

Output ensembling does not appear in Table 11. It does not train either forecaster: the diffusion backbone and the external model stay fixed, and only a mixture weight is selected by validation CRPS.

Table 11: CRPS for fine-tuned and frozen-backbone input concatenation and FiLM. Fine-tuned entries match Table 1. Best values are **bold**.

	Input concatenation		FiLM	
Dataset	Fine-tuned	Frozen backbone	Fine-tuned	Frozen backbone
Electricity	**0.062±0.000**	0.065±0.000	**0.054±0.000**	0.060±0.000
Exchange	**0.007±0.000**	0.009±0.000	**0.008±0.000**	0.010±0.000
KDDCup	**0.289±0.002**	0.332± 0.001	**0.268±0.005**	0.292± 0.001
M4-Hourly	**0.027±0.001**	0.034 ± 0.001	**0.026±0.001**	0.040 ± 0.001
Solar	**0.308±0.004**	0.390±0.005	**0.360±0.002**	0.423 ±0.003
Traffic	**0.117±0.000**	0.120±0.001	**0.116±0.001**	0.119±0.000
UberTLC	**0.187±0.001**	0.190 ± 0.000	**0.184±0.001**	0.189 ± 0.000
Wikipedia	**0.241±0.001**	0.246±0.001	**0.235±0.001**	0.245±0.001

B.8 Additional Generative Baseline Protocols

We adapt the authors' K^{2}VAE and CN-Diff implementations (Wu et al., 2025; Rishi et al., 2025) to one target channel, using Control-TS's GluonTS splits, context/horizon lengths, rolling test windows, and per-window mean-absolute scaling. These are task-matched univariate evaluations, not reproductions of the original multivariate benchmarks.

Both use Adam and 100 batches/epoch. Checkpoints and hyperparameters are selected by validation CRPS, evaluated each epoch with 16 paths (four for Traffic/Wikipedia); testing uses 100 paths.

K^{2}VAE. We retain reversible normalization and use 50 epochs, gradient clipping 0.5, and batch size 32 for Electricity, 64 otherwise. We compare a globally learned Koopman operator with a combined formulation that adds a local least-squares estimate, using learning rates $\{10^{-3}, 10^{-4}\}$. Validation selects the global-only variant throughout and 10^{-3} for M4-Hourly/Wikipedia, 10^{-4} otherwise. Some combined-operator runs are unstable with one target channel. KDDCup/M4-Hourly/UberTLC use the released Electricity architecture. Patch lengths are 10 for Exchange/Wikipedia and eight otherwise, dividing context and horizon lengths.

CN-Diff. We retain last-observation centering and use up to 100 epochs, batch size 64, learning rate 10^{-3}, and early-stopping patience 10. The validation-CRPS plateau scheduler uses factor 0.5 and patience five. Searching widths $\{64, 256\}$ selects 256 for M4-Hourly/Solar, 64 otherwise. Sampling uses 100 steps with the released quadratic schedule. Changes from upstream are the 100-batch epoch cap, plateau scheduler, and 100 rather than 10 test paths. We also initialize the previously uninitialized temporal-layer weights and biases in the nonlinear transformation T_ϕ, using the same routine as for its feature layer.

B.9 TSDiff Refinement Protocol

We apply the official TSDiff refinement procedure (Kollovieh et al., 2023) to the frozen backbone from which Control-TS is trained, initialized by the DeepAR checkpoints that provide its DeepAR controls. For each test window, 100 DeepAR samples are refined for 20 iterations by likelihood maximization (ML-Q; learning rate 0.1) or Langevin sampling (LMC-Q; step size 0.1, noise scale 0.1), both with the quantile regularizer. The diffusion step used as the density surrogate is chosen by the original heuristic on one training batch. Both variants start from the same forecasts, no model parameters are updated, and hyperparameters follow the released configuration. Evaluation uses the same test windows, 100 forecast paths, and CRPS computation as Control-TS. We report mean$\pm$SD over three sampling seeds. TFT is not used because the procedure refines sample paths, whereas TFT outputs quantiles.

B.10 Per-Variant Forecasting Results

For each dataset and quantile set (Q1, Q3, or Q5), Table 2 reports the test CRPS of Control-TS using the external forecaster selected by validation CRPS. These validation-selected configurations need not achieve the lowest test CRPS in Table 12. Here we provide the complete variant-level CRPS for all six configurations: Control-TS-Q1, Control-TS-Q3, and Control-TS-Q5, each instantiated with either DeepAR or TFT as the external probabilistic forecaster (Section 2.4, Appendix A.3). All other components are kept fixed to the configurations described in Appendix B.4, and the TSDiff baseline is included for reference.

Table 12 shows how test CRPS varies with the external forecaster and quantile granularity. TFT-based variants attain the lowest test CRPS on Electricity, Exchange, Solar, and UberTLC, and DeepAR-based variants on KDDCup, M4-Hourly, and Traffic; the two forecasters tie on Wikipedia. On Solar, TFT-Q5 achieves a test CRPS 0.001 lower than DeepAR-Q5, but validation CRPS selects DeepAR-Q5. On KDDCup, validation selects TFT-Q3 (0.249) although DeepAR-Q5 attains 0.245. On every dataset, a multi-quantile variant (Q3 or Q5) attains the lowest test CRPS, with Q1 tying on Exchange, consistent with the granularity discussion in Section 3.3.

Table 12: Per-variant CRPS on the eight benchmark datasets for each external forecaster (rows) and quantile variant Q1/Q3/Q5 (Section 2.4), with the TSDiff baseline for reference. The best entry per column is shown in **bold** and the second-best is underlined. Tied entries are marked jointly.

External forecaster	Variant	Electricity	Exchange	KDDCup	M4-Hourly	Solar	Traffic	UberTLC	Wikipedia
Baseline		0.049	0.012	0.312	0.034	0.402	0.099	0.172	0.211
DeepAR	Q1	0.054	0.009	0.303	0.025	0.419	0.098	0.177	0.203
	Q3	0.057	0.009	0.267	0.026	0.397	**0.097**	0.171	0.205
	Q5	0.056	0.009	**0.245**	**0.022**	0.341	0.100	0.173	**0.200**
TFT	Q1	0.049	**0.008**	0.282	0.028	0.408	0.107	0.170	0.204
	Q3	0.048	**0.008**	0.249	0.026	0.392	0.102	**0.169**	0.202
	Q5	**0.047**	**0.008**	0.256	0.025	**0.340**	0.101	0.172	**0.200**

B.11 Observed Context as Control

Both sides use the same full-depth residual architecture. The Control-TS entries are the Q1, Q3, and Q5 rows of Table 2. Context only repeats the observed context once, three times, or five times, matching that channel count. It includes observed historical lag channels, zeros all forecast-horizon input channels, and does not load or call an external forecaster. The Solar Q1 result appears to be an outlier, but Control-TS with Q5 still achieves lower CRPS.

Table 13: **Observed context as control.** CRPS (mean±SD) using the same full-depth residual architecture. Control-TS values are from Table 2. The better value in each pair is **bold**, including ties.

	Q1		Q3		Q5	
Dataset	Control-TS	Context $\times 1$	Control-TS	Context $\times 3$	Control-TS	Context $\times 5$
Electricity	0.049±0.000	**0.047±0.000**	**0.048±0.000**	0.049±0.000	**0.047±0.000**	0.048±0.000
Exchange	**0.008±0.000**	0.010±0.000	**0.008±0.000**	0.011±0.000	**0.008±0.000**	0.012±0.000
KDDCup	**0.282±0.001**	0.311±0.001	**0.249±0.001**	0.319±0.001	**0.256±0.001**	0.282±0.001
M4-Hourly	**0.025±0.000**	0.045±0.000	**0.026±0.000**	0.033±0.000	**0.022±0.000**	0.030±0.000
Solar	0.408±0.000	**0.375±0.000**	**0.392±0.000**	0.449±0.000	**0.341±0.000**	0.440±0.001
Traffic	**0.098±0.000**	0.100±0.000	**0.097±0.000**	0.099±0.000	0.100±0.000	**0.099±0.000**
UberTLC	**0.170±0.000**	0.171±0.001	**0.169±0.000**	0.176±0.001	**0.172±0.000**	0.180±0.001
Wikipedia	**0.203±0.000**	0.212±0.000	**0.202±0.000**	0.209±0.000	**0.200±0.000**	0.206±0.000

B.12 Architectural Ablations

This subsection evaluates three design choices in Control-TS: **(i)** zero-initialized channel projection operators, **(ii)** the initialization of the Controller residual blocks, and **(iii)** depth-matched residual injection through all three Controller blocks. Unless otherwise stated, all ablated variants use the same frozen diffusion backbone, training protocol, external forecaster family, and quantile granularity as the corresponding Control-TS configuration selected by validation CRPS for each dataset (Table 2). We report CRPS on the full test set.

Projection initialization. Table 14 compares the default Control-TS model against a full-depth variant without zero-initialized channel projections. Removing zero-initialization degrades performance on all eight datasets and is particularly harmful on Solar, Wikipedia, M4-Hourly, and KDDCup, consistent with the role of zero-initialized projections in preserving the pretrained backbone behavior at the start of training and introducing control gradually.

Table 14: Projection-initialization ablation for full-depth Control-TS. Best values are **bold**.

Dataset	Control-TS	No zero-init
Electricity	**0.047**	0.051
Exchange	**0.008**	0.009
KDDCup	**0.249**	0.302
M4-Hourly	**0.022**	0.027
Solar	**0.341**	0.433
Traffic	**0.097**	0.113
UberTLC	**0.169**	0.184
Wikipedia	**0.200**	0.250

Controller initialization. For completeness, we compare random initialization of the controller residual blocks (Control-TS) with initialization from the corresponding pretrained backbone weights, keeping the remaining setup unchanged. Table 15 reports the comparison.

Table 15: Controller-initialization ablation for full-depth Control-TS (CRPS; lower is better). Control-TS initializes its controller residual blocks randomly. Pretrained initialization instead starts each controller block from a copy of the corresponding residual block of the pretrained diffusion backbone; the copies are trained while the backbone remains frozen, and all other settings are unchanged.

Dataset	Pretrained initialization	Control-TS
Electricity	0.050±0.000	0.047±0.000
Exchange	0.009±0.000	0.008±0.000
KDDCup	0.305±0.000	0.249±0.001
M4-Hourly	0.030±0.000	0.022±0.000
Solar	0.401±0.000	0.341±0.000
Traffic	0.107±0.001	0.097±0.000
UberTLC	0.175±0.000	0.169±0.000
Wikipedia	0.220±0.000	0.200±0.000

Controller-depth ablations. Table 16 studies whether residual corrections need to be injected throughout the full backbone depth. We compare the default three-block Controller against shallower variants that inject residual corrections only through block 0, or through blocks 0 and 1. The default three-block model has the lowest or tied-lowest displayed CRPS on all eight datasets; Exchange, the two-block variant ties it at 0.008 after rounding. Among the shallow variants, using two Controller blocks is generally better than using one, indicating that control benefits from being injected beyond the earliest layer. Overall, these results support the use of depth-matched residual control.

Table 16: Controller-depth ablations. **bold** identifies the full-depth reference, which is lowest or tied-lowest at displayed precision.

Dataset	1 block	2 blocks	Control-TS
Electricity	0.052	0.052	**0.047**
Exchange	0.009	0.008	**0.008**
KDDCup	0.286	0.285	**0.249**
M4-Hourly	0.032	0.026	**0.022**
Solar	0.444	0.415	**0.341**
Traffic	0.125	0.118	**0.097**
UberTLC	0.203	0.193	**0.169**
Wikipedia	0.248	0.245	**0.200**

B.13 Robustness to Degraded Control Signals

To complement the representative robustness results shown in the main text, we report here the full evaluation across all eight datasets and all tested perturbation levels, including the more extreme setting $\sigma = 0.1$. As in the main-text experiment, we perturb the quantile-based control signal at inference time by adding Gaussian noise in normalized control space, while keeping both the trained Controller and the frozen diffusion backbone fixed. We then evaluate CRPS on the full test set for each dataset.

Table 17 reports the resulting mean CRPS values under increasing noise levels, together with the unguided diffusion-backbone baseline. Entries better than the backbone are **bold**; ties are underlined. The degradation is clearly dataset-dependent. KDDCup is the most robust: Control-TS remains better than the backbone at every tested noise level, including the stress-test setting $\sigma = 0.1$. Wikipedia is similarly stable, and ties the backbone even at the strongest perturbation. Solar also degrades gradually and remains better than the backbone up to $\sigma = 0.05$, crossing above it only at $\sigma = 0.1$. In contrast, Electricity and M4 are more sensitive: both remain improved under mild perturbations ($\sigma = 0.01$ and $\sigma = 0.02$), but fall below the backbone at $\sigma = 0.05$ and deteriorate further at $\sigma = 0.1$. Traffic and UberTLC show a similar but milder pattern, staying competitive under small perturbations before losing their advantage under stronger corruption. Exchange is the most fragile among the eight datasets: although Control-TS remains better than the backbone at $\sigma = 0.01$, the gain disappears already at $\sigma = 0.02$ and degrades substantially thereafter. Taken together, these results show that

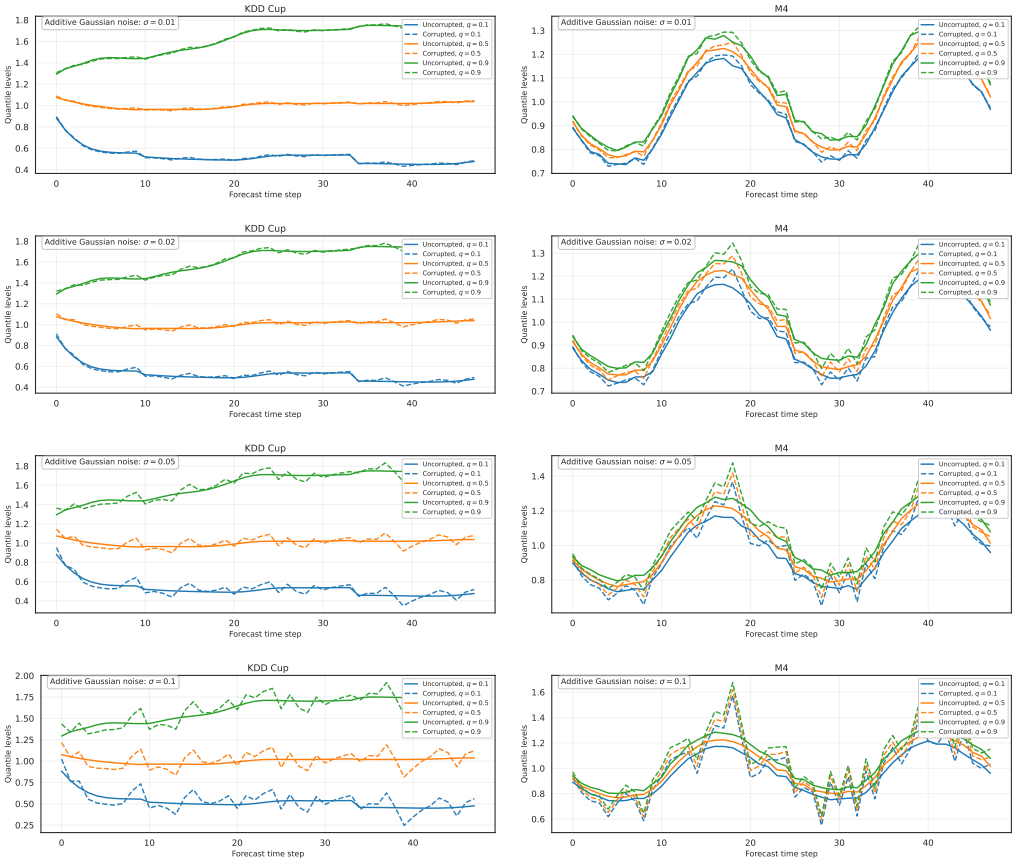


Figure 7: Examples of corrupted quantile-control signals under additive Gaussian perturbations in normalized control space. Solid curves show the original quantile trajectories and dashed curves show the perturbed controls for representative datasets and increasing noise levels.

Control-TS is generally tolerant to mild noise in the control signal, but that stronger degradation can negate its benefit in a distinctly dataset-dependent manner.

Figure 7 visualizes the effect of Gaussian perturbations on the quantile-based control signal for two representative datasets, KDDCup and M4, across noise levels $\sigma \in \{0.01, 0.02, 0.05, 0.1\}$. In each panel, the corrupted quantile trajectories are shown against their uncorrupted counterparts for $q = 0.1$, 0.5, and 0.9. As expected, the perturbation is mild at small noise levels and becomes progressively more disruptive as σ increases, eventually distorting both the central trajectory and the uncertainty envelope. This visualization complements Table 17 by showing directly how Gaussian degradation weakens the uncertainty scaffold provided to the Controller.

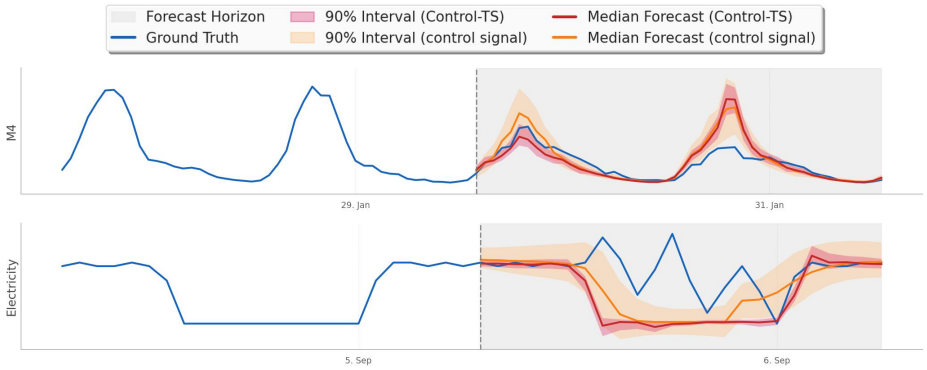


Figure 8: Failure cases of Control-TS: misaligned quantile scaffolds where the forecast follows the unreliable signal.

Table 17: CRPS under Gaussian degradation of the control signal at inference time. Noise is added in normalized control space. Entries better than the unguided diffusion-backbone baseline are shown in **bold**, while ties are underlined.

Dataset	Clean	$\sigma = 0.01$	$\sigma = 0.02$	$\sigma = 0.05$	$\sigma = 0.1$	Backbone
Electricity	**0.047**	**0.048**	0.049	0.058	0.080	0.049
Exchange	**0.008**	**0.010**	0.014	0.031	0.065	0.012
KDDCup	**0.249**	**0.249**	**0.251**	**0.252**	**0.261**	0.312
M4	**0.022**	**0.023**	**0.024**	0.037	0.053	0.034
Solar	**0.341**	**0.374**	**0.377**	**0.389**	0.424	0.402
Traffic	**0.097**	**0.098**	0.099	0.108	0.131	0.099
UberTLC	**0.169**	**0.170**	**0.169**	0.174	0.182	0.172
Wikipedia	**0.200**	**0.201**	**0.201**	**0.203**	0.211	0.211

B.14 Failure Cases under Misaligned Control Signals

Beyond the controlled Gaussian perturbations of Appendix B.13, we also examine naturally occurring cases in which the external forecaster's quantile scaffold is misaligned with the realized future. While the diffusion backbone typically buffers Control-TS against such misalignment (Section 3.4, Figure 4b), Figure 8 shows representative cases where the controller does follow a misaligned scaffold and the resulting forecast drifts toward the unreliable signal. These cases are uncommon in our experiments but are reported for transparency and to bound the practical reliability of residual control under in-the-wild miscalibration of the external forecaster.

B.15 Computational Overhead: Full Accounting

This appendix expands the analysis of Section 3.6. We describe the measurement protocol, the parameter budget of Control-TS relative to the frozen diffusion backbone, the structural reasons inference cost shows modest growth while training cost is more pronounced, and report the per-dataset values for all eight benchmarks corresponding to Table 19.

Protocol. For each of the eight benchmarks we load checkpoints for both the frozen backbone and the trained Control-TS, build one batch of size B=16 from the standard test split, and measure on a single NVIDIA RTX 2080 Ti: **(i)** end-to-end sampling wall-clock, FLOPs, and peak VRAM for one forecast call (T_{diff}=100 reverse-diffusion steps with quantile observation self-guidance, see Appendix A.1); and **(ii)** wall-clock, FLOPs, and peak VRAM for one denoising training step (forward pass, backward pass, and optimizer update). Wall-clock is averaged over 5 inference runs after 5 warm-up runs and over 100 training-step runs after 10 warm-up runs. FLOPs are obtained via `torch.profiler` with FLOP attribution enabled. Peak VRAM is the maximum allocated GPU memory at a given moment during the same call.

Parameter accounting. A trained Control-TS model comprises three components: **(a)** the frozen backbone, whose parameters θ_B are held fixed throughout Controller training; **(b)** a trainable Controller branch, consisting of a residual stack matching the backbone's architecture together with a Controller input projection E_{θ_C} acting on the control signal c; and **(c)** zero-initialized 1×1 channel projection operators $\{Z^{(k)}_{\theta_C}\}_{k=1}^{K}$ that gate residual injection at each backbone depth (see Section 2.3 and Appendix A.2). At training, only the Controller branch and its zero-initialized projections receive gradient updates; the diffusion backbone is held fixed. The Controller branch is therefore approximately the same size as the backbone but slightly smaller, because it does not include the backbone's input/output projections, time-embedding head, or lag-feature head. Per-dataset parameter counts are reported in Table 18.

Table 18: Parameter budget of Control-TS per dataset. The diffusion backbone is held fixed during Controller training; only the Controller branch receives gradient updates. Counts in thousands of parameters (k); ratio is Controller / Backbone.

Dataset	Backbone (frozen) [k]	Controller (trainable) [k]	Controller / Backbone
Electricity	193.4	191.4	0.99
Exchange	193.4	190.1	0.98
KDDCup	193.4	190.0	0.98
M4	192.1	188.2	0.98
Solar	193.4	191.4	0.99
Traffic	193.4	190.0	0.98
UberTLC	193.4	190.0	0.98
Wikipedia	191.9	188.0	0.98

Why inference cost modestly grows. At each reverse-diffusion timestep t, the sampler performs two forward passes through the denoising network: an unconditional reverse-kernel mean (Eq. 8), evaluated with the controlled denoiser $\hat{\epsilon}_\theta(x_t, t \mid c)$ for Control-TS, and an observation self-guidance score $-\nabla_{x_t} E$ (Eq. 11), which differentiates through the backbone-only Tweedie one-step estimate $\tilde{x}_{0,B}$ (Eq. 7, evaluated with the unmodified backbone) with respect to the noisy sample x_t. For both models, the self-guidance pass is evaluated through the unmodified diffusion backbone and accounts for roughly two-thirds of per-step compute in our profiler splits. Only the unconditional reverse-kernel mean differs across models. Because the Controller mirrors only the residual block stack of the backbone (and not its input/output projections, time-embedding head, or lag-feature head), this slice grows by $\sim 1.2\times$-$1.7\times$, leading to a per-step total ratio of $\sim 1.07\times$-$1.25\times$ over $T_{\text{diff}}{=}100$ steps–matching the inference FLOPs columns of Table 19.

Why inference VRAM gap is small. At inference, all diffusion parameters are frozen (`requires_grad`=False), so the autograd tape used to evaluate the self-guidance score retains only the activations needed for the input gradient $\partial E/\partial x_t$, not weight gradients. Since this score is the architectural bottleneck of the reverse-diffusion step and is computed through the unmodified backbone, peak inference VRAM is set by the same self-guidance tape for both models, and only a small Controller forward-only contribution from the unconditional reverse-kernel mean is added on top. Peak inference VRAM therefore grows by less than 7% across all datasets.

Why training cost grows more. Training requires gradients through both the controlled denoising network and the noise-prediction loss, retaining activations through both backbone (frozen) and Controller. Across datasets, peak training VRAM rises by a consistent 64% to 66% as the Controller's activation tape is added to the existing backbone tape. Training-step wall-clock stays below 15% on six of eight datasets and below 66% on all of them, so the per-step training budget never doubles. To isolate the Controller's intrinsic forward+backward cost, the timing window excludes the per-step regeneration of control quantiles (constructed once outside the timing window): in training-mode, control quantiles are cached upstream rather than regenerated per step, so including the GluonTS `predict` path inside the window would inflate per-step time with framework overhead unrelated to the Control-TS forward/backward. Per-step training FLOPs from the profiler are reported alongside.

Full per-dataset table. Table 19 reports all six measurements for all eight benchmarks. Inference overhead is small and bounded across datasets (17% to 61% wall-clock, 7% to 25% FLOPs, 2% to

7% peak VRAM), while training overhead is consistent in VRAM (64% to 66%) and bounded in wall-clock–under 20% on six of eight benchmarks and under 66% on all of them.

Table 19: Computational overhead of Control-TS relative to the frozen backbone for all eight benchmarks (B=16, T_{diff}=100, with quantile observation self-guidance). Each Δ column is the relative increase (%) of Control-TS over the diffusion backbone; GFLOPs columns report Backbone / Control-TS.

	Inference (sampling)			Training (one step)		
Dataset	ΔTime (%)	GFLOPs	ΔVRAM (%)	ΔTime (%)	GFLOPs	ΔVRAM (%)
Electricity	+19.1	21.5 / 26.8	+6.1	+13.5	0.20 / 0.24	+66.2
Exchange	+61.1	24.2 / 28.7	+4.0	+65.6	0.23 / 0.24	+66.0
KDDCup	+17.1	21.5 / 25.2	+5.9	+13.2	0.20 / 0.21	+64.8
M4	+39.2	17.0 / 18.2	+3.5	+6.5	0.17 / 0.14	+63.9
Solar	+21.4	21.5 / 26.4	+4.5	+14.9	0.20 / 0.24	+65.0
Traffic	+21.0	21.5 / 24.8	+4.2	+10.6	0.20 / 0.21	+64.3
UberTLC	+30.3	21.5 / 25.2	+5.9	+14.7	0.20 / 0.21	+64.8
Wikipedia	+31.5	18.4 / 20.1	+2.4	+47.3	0.18 / 0.16	+65.7

Take-away. Self-guidance dominates inference cost and is shared between the frozen diffusion backbone and Control-TS, so the Controller adds only a bounded slice of compute and memory at sampling time. Training is where the Controller is fully paid for, but only as a one-time amortized cost during Controller fitting, after which a Control-TS model is queried at inference at near-baseline cost. This profile makes residual control attractive whenever a pretrained diffusion forecaster is already deployed: the Controller can be fit and shipped alongside the backbone with only a marginal sampling-time penalty.

B.16 CLPS Details: THTR Construction and Linear Evaluator

We compute CLPS under a Train-on-Hybrid, Test-on-Real (THTR) evaluation using $N_{\text{hyb}} = 20{,}000$ hybrid samples, where "hybrid" refers to real observed contexts paired with synthetic forecast horizons sampled from the model. We hold the training contexts fixed across TSDiff and Control-TS, so only the generated futures differ, with no test leakage into the hybrid data. We sample N_{hyb} real contexts $\mathbf{y}_{\text{ctx}}$ from the training set and generate one synthetic target horizon per context by running model inference conditioned on $\mathbf{y}_{\text{ctx}}$ (and its corresponding control signal c), yielding hybrid pairs $(\mathbf{y}_{\text{ctx}}, \hat{\mathbf{y}}_{\text{tar}})$. We then fit a closed-form ridge forecaster on these hybrid pairs and evaluate it on the real test set using CRPS. Because the linear evaluator is deterministic, it produces a single forecast trajectory for each test context. To use the same CRPS evaluation pipeline as the generative models, we replicate this trajectory across the 100 sampled paths used for CRPS computation; these draws are therefore identical and introduce no additional Monte Carlo variability. For large N_{hyb}, drawing one synthetic target per context yields N_{hyb} i.i.d. hybrid samples from the joint distribution

$$p_{\text{hyb}}(\mathbf{y}_{\text{ctx}}, \hat{\mathbf{y}}_{\text{tar}}) = p_{\text{data}}(\mathbf{y}_{\text{ctx}})\, p_{\text{gen}}(\hat{\mathbf{y}}_{\text{tar}} \mid \mathbf{y}_{\text{ctx}}),$$

and Monte Carlo estimation error for statistics of these pairs decays as $O\left(\frac{1}{\sqrt{N_{\text{hyb}}}}\right)$ (Owen, 2013), which is adequate for the stability of the closed-form linear evaluator used by CLPS.

B.17 Combined Additional-Backbone Scope Study

Paired CSDI–DeepAR and MG-TSD–DeepVAR evaluations use matched windows and 100 paths (Table 20). CSDI uses CRPS; multivariate MG-TSD uses CRPS-sum, scoring the sum across variates.

Integration. Both keep native forecasting, with frozen backbones and external forecasters. Quantile controls are the Controller's input; they are not concatenated into CSDI's side information or into MG-TSD's recurrent history encoders. The Controller adds a zero-initialized projection at the input of each

corresponding backbone residual block. CSDI uses four 64-channel residual layers, eight attention heads, and no feature-axis attention (univariate). MG-TSD uses eight 8-channel residual blocks shared across granularities 1 and 4; its two-layer 128-unit recurrent encoders remain frozen backbone modules, and control is applied at each autoregressive step. Q5 controls use $\{0.1, 0.25, 0.5, 0.75, 0.9\}$ from 100 external paths, with predicted horizon quantiles appended to the repeated scaled context and aligned to the forecast origin and backbone scale. For MG-TSD's coarser granularity, the horizon control is the quantile of DeepVAR paths after averaging each coarse time block, not the average of per-step quantiles.

Table 20: Paired test-score reductions (%) from Control-TS, using 100 forecast paths. Positive values indicate improvement; negative values indicate regression.

	CSDI: univariate	MG-TSD: multivariate
Dataset	CRPS	CRPS-sum
Electricity	$+7.90$	$+6.35$
Exchange	$+9.98$	-4.16
KDDCup	$+3.90$	-7.23
M4-Hourly	$+9.73$	$+3.98$
Solar	$+0.34$	$+38.27$
Traffic	$+11.05$	$+1.91$
UberTLC	$+1.58$	-2.29
Wikipedia	$+2.67$	$+8.48$

CSDI training. Backbone/Controller: Adam 10^{-3}, batch size 64, up to 100 epochs of 100 batches, early-stopping patience 10, plateau factor 0.5/patience five, and validation-CRPS selection. DeepAR: fitting-only data, 15 epochs of 100 batches, two 40-unit layers, dropout 0.1, batch size 32, and validation negative-log-likelihood selection. Sampling uses all 50 ancestral steps, with quadratic noise from 10^{-4} to 0.5.

MG-TSD training. Both use 30 epochs of 100 updates; effective batch size is 128 for Solar, 32 otherwise. The baseline uses OneCycle (peak 0.01); the Controller uses fixed-rate Adam 10^{-3}, clipping 0.5. Validation CRPS-sum at epoch 1 and every five epochs selects checkpoints. Sampling: 100 steps, linear noise ending at 0.1, sharing ratios $\{1, 0.6\}$, loss weights $\{0.9, 0.1\}$. DeepVAR: two 128-unit LSTM layers, dropout 0.1, rank-10 Gaussian output (eight for Exchange), 200 epochs of 100 updates, Adam 10^{-3}, marginal Gaussian CRPS; validation selects fixed-rate/plateau candidates.

Splits and evaluation. Fitting excludes validation/test targets. CSDI uses reference context/horizon lengths and seven validation windows for Electricity/Solar/Traffic, five for Exchange/Wikipedia, one otherwise. MG-TSD reserves three validation horizons; context equals horizon. Dimensions/horizons follow Table 5's training-series counts/prediction lengths. Wikipedia fitting ends before the first test origin. MG-TSD panels use calendar time, except relative-origin alignment for KDDCup/M4-Hourly/UberTLC.

B.18 Additional Qualitative Examples

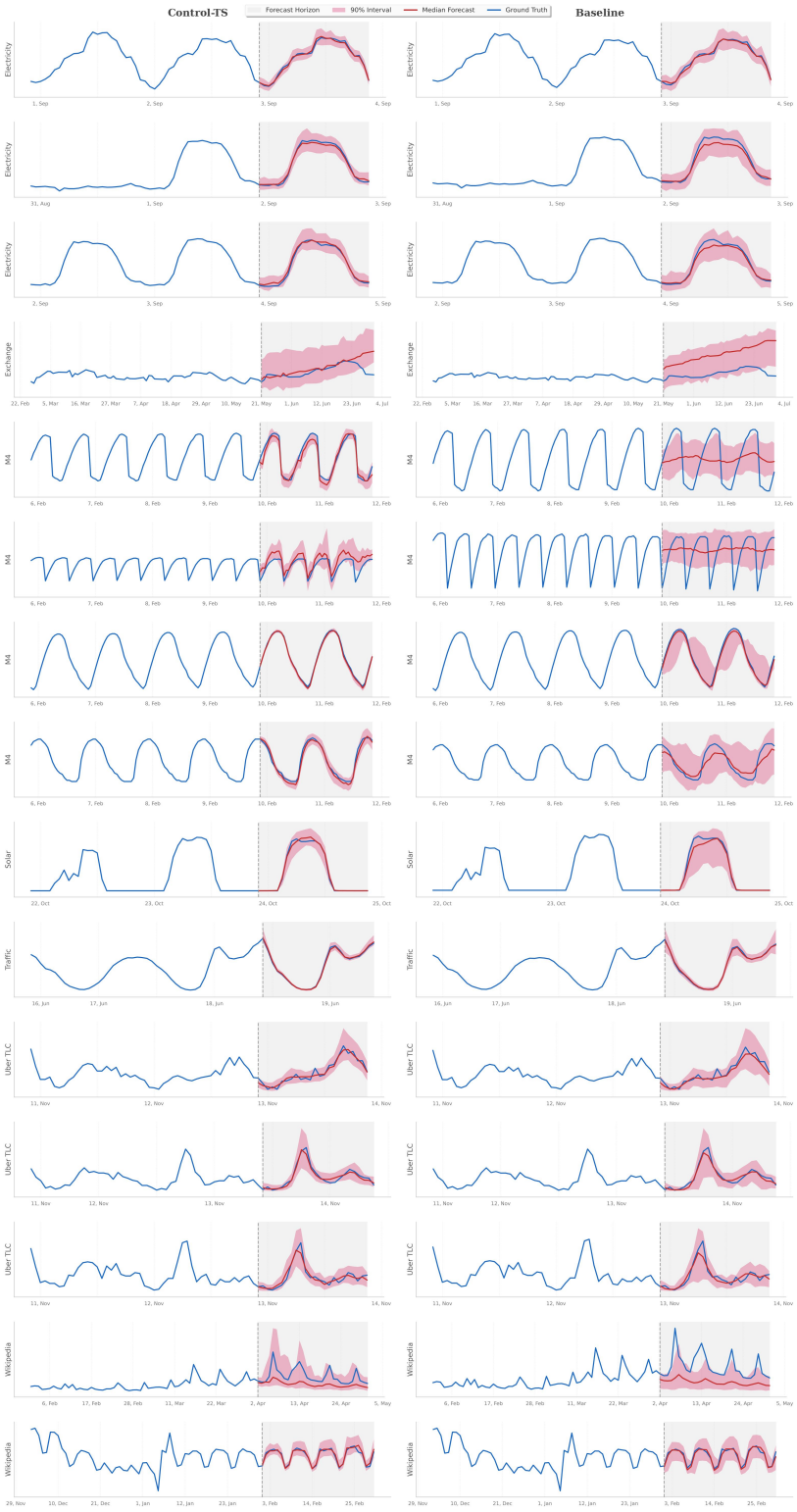


Figure 9: Additional qualitative comparisons between Control-TS and the TSDiff baseline across selected benchmark series.

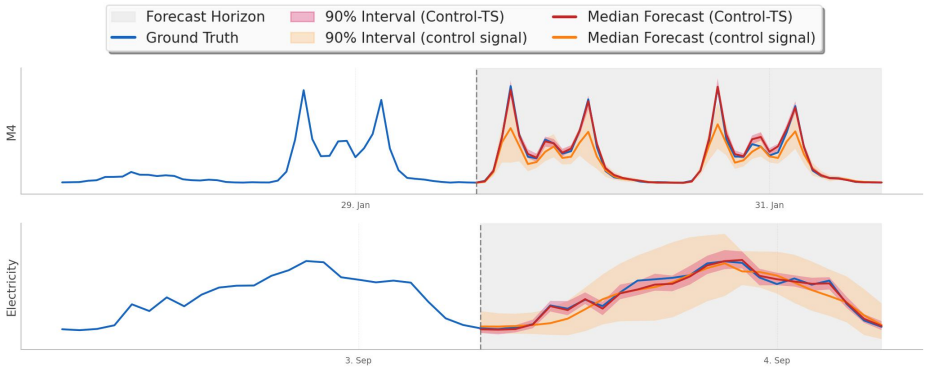


Figure 10: Examples with accurate quantile scaffolds: Control-TS tightens its predictive distribution around the realized future.