OVERCOMING THE MODALITY GAP IN CONTEXT-AIDED FORECASTING*

Vincent Zhihao Zheng
ServiceNow AI Research
McGill University

Étienne Marcotte
ServiceNow AI Research
etienne.marcotte@servicenow.com

Arjun Ashok
ServiceNow AI Research
Université de Montréal
Mila - Quebec AI Institute

Andrew Robert Williams
ServiceNow AI Research
Université de Montréal
Mila - Quebec AI Institute

Lijun Sun
McGill University

Alexandre Drouin
ServiceNow AI Research
Mila - Quebec AI Institute

Valentina Zantedeschi
ServiceNow AI Research

ABSTRACT

Context-aided forecasting (CAF) holds promise for integrating domain knowledge and forward-looking information, enabling AI systems to surpass traditional statistical methods. However, recent empirical studies reveal a puzzling gap: multimodal models often fail to outperform their unimodal counterparts. We hypothesize that this underperformance stems from poor context quality in existing datasets, as verification is challenging. To address these limitations, we introduce a semi-synthetic data augmentation method that generates contexts that are both descriptive of temporal dynamics and verifiably complementary to numerical histories. This approach enables massive-scale dataset creation, resulting in **CAF-7M**, a corpus of 7 million context-augmented time series windows, including a rigorously verified test set. [1] We demonstrate that semi-synthetic pre-training transfers effectively to real-world evaluation, and show clear evidence of context utilization. Our results suggest that dataset quality, rather than architectural limitations, has been the primary bottleneck in context-aided forecasting.

Track: Research

1 INTRODUCTION

Time series forecasting is a core component of decision-making across industries, from supply chain optimization to financial planning. While statistical methods like ARIMA (Hyndman & Athanasopoulos, 2018) have been workhorses for decades, *context-aided forecasting* (CAF) has emerged as a promising new paradigm, where external context, often in the form of unstructured textual information (e.g., incident reports, operational notes, expert narratives), is integrated with the numerical histories (Liu et al., 2024a; Williams et al., 2025; Wang et al., 2025). Such context may include domain knowledge and forward-looking factors (e.g., disruptions, policy changes, campaigns) that are not inferable from historical numerical data alone, yet are essential for accurate predictions.

Despite its enormous potential, recent empirical analyses show that multimodal forecasting models that fuse time series with textual context frequently *fail to outperform unimodal baselines* (Zhang et al., 2025). We hypothesize that this performance gap stems from low context usefulness in both training and test CAF datasets. For a piece of context to be useful, it should be *descriptive* of

*Extended version available at https://arxiv.org/abs/2603.12451.

[1] **CAF-7M** is publicly available at https://huggingface.co/datasets/ServiceNow/CAF_7M. The code for the **DoubleCast** model is available at https://github.com/ServiceNow/DoubleCast.

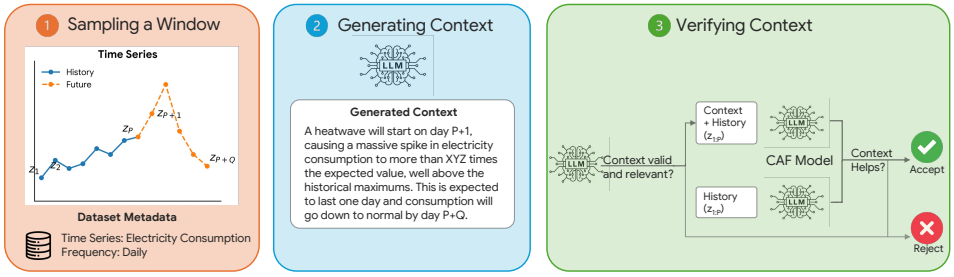


Figure 1: The data-augmentation pipeline: (1) From each source dataset, we sample forecasting windows consisting of a numerical history and prediction horizon, along with dataset metadata. (2) We generate scenario-style textual context conditioned on the window. (3) We verify context relevance by checking whether a strong CAF method achieves better predictions with context than without (e.g., lower CRPS); windows are *accepted* if it does and *rejected* otherwise.

the underlying dynamics, *aligned* with the temporal window of interest, and *complementary* to patterns already visible in the numerical trajectory. However, obtaining high-quality data at the scale required for training AI models often relies on matching time series with web-scraped text (e.g., news articles) (Liu et al., 2024b; Wang et al., 2024; 2025), whose predictive utility remains unverified. Without datasets where context is *provably useful*, progress in CAF remains bottlenecked.

To overcome this modality gap, we introduce a context-generation methodology that enables researchers to convert any numerical time series dataset into a context-aided forecasting dataset with provably useful context. Our methodology consists of two phases: generation and verification. In the generation phase, we prompt a Large Language Model (LLM) to generate plausible scenarios that explain differences in dynamics between the historical and future portions of a time series window. In the verification phase, we task a state-of-the-art context-aided forecasting model with predicting the future given the history and generated context, keeping only window-context pairs for where the forecast improves with context, compared to without it. This ensures that retained contexts are descriptive and complementary to numerical histories.

Contributions.

- A semi-synthetic **methodology for generating verifiably useful contexts** from any time series dataset, enabling massive-scale and high-quality CAF data.
- **CAF-7M**: a corpus of 7 million context-augmented time series windows spanning 11 domains, including a rigorously verified test set of 904 windows.
- Evidence that our methodology enables **context utilization** and **simulated-to-real transfer**, via a multimodal architecture trained exclusively on CAF-7M.

We validate CAF-7M's effectiveness by training DoubleCast, a newly introduced architecture which augments the Chronos foundation model (Ansari et al., 2024) with CAF capabilities (see App. C). When trained on CAF-7M, DoubleCast (i) effectively leverages context, (ii) achieves performance comparable to state-of-the-art CAF models while being significantly more cost-effective, and (iii) consistently outperforms Chronos, its unimodal counterpart. Further, DoubleCast outperforms both unimodal baselines and state-of-the-art models on the real-world ChatTime benchmark (Wang et al., 2025), demonstrating that training on semi-synthetic data transfers to real-world CAF. This work highlights that meaningful progress in context-aided forecasting benefits from benchmarks that explicitly have context complementarity, and that architectures enforcing explicit dual-modality alignment can learn to use context even when training data contain noisy textual signals.

2 CAF-7M: A Context-Aided Forecasting Dataset

We aim for context that is both plausible and informative, rather than trivially describing future values in textual form. As shown in Fig. 1, our pipeline selects a time series dataset as a starting point, generates plausible context, and filters for quality.

Generating Plausible Contexts Large language models are accurate context-aided forecasters even without specific training (Williams et al., 2025), suggesting they can also generate context for a given window. We do so by prompting the model with both historical and future values, along with a short description of the time series domain. We then instruct the model to produce a scenario that would help predict the future values, focusing in particular on changes in dynamics from the historical series. To address limited diversity in LLM outputs, we prompt the model to avoid repeating scenarios from previously generated contexts (Merrill et al., 2024). See App. B.1 for the prompt.

Ensuring Informative Contexts LLM-generated contexts may fail to introduce new information, contain incorrect information, or be too difficult for state-of-the-art CAF models to exploit. To ensure quality, we first perform a basic check by prompting an LLM to verify whether the context explicitly mentions the variable represented in the time series and whether the scenario is plausible for that variable. See App. B.1 for the prompt used.

We then filter contexts by checking their compliance with Eq. (2), i.e., does including the context improve the forecast? We use a Direct Prompt forecaster (Williams et al., 2025) with a strong LLM (e.g., GPT-5.2) as the distribution estimator, as it has been shown to perform well both with and without context. We use the Continuous Ranked Probability Score (CRPS; Gneiting & Raftery (2007)) as the proper scoring rule. Samples for which the CRPS with context exceeds the CRPS without context are discarded, considering that the context may be insufficiently informative. The rest of the samples for which the forecast of the model improves with the generated context are kept. This provides a set of contexts that are complementary to numerical histories, accounting for dynamic shifts and providing external predictive cues.

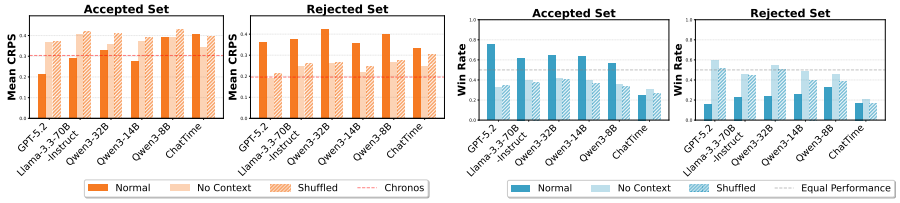
CAF-7M We now introduce CAF-7M (**C**ontext-**A**ided **F**orecasting), a large-scale context-aided forecasting corpus built using this methodology. CAF-7M contains 7,433,239 context-augmented time series windows for training and 904 for testing. We sample the numerical data for these windows from 65 real-world datasets spanning diverse domains, with 40 used for training and 25 reserved for testing (see Tab. 3). We follow the dataset selection and train/test split from Chronos (Ansari et al., 2024) to ensure domain diversity and comparability. App. B.2 specifies the number of windows from each dataset. Prediction horizons are set according to data frequency (e.g., 64 for hourly and daily; see App. B.1), and we sample history lengths uniformly between the prediction horizon and 512. Comparisons with previously datasets are done in App. A.2 and Tab. 2.

We then augment the raw numerical time series windows with synthetic contexts. We use `Llama-3.3-70B-Instruct` (Grattafiori et al., 2024) to generate contexts and `GPT-5.2` (OpenAI, 2026) to verify the contexts with the Direct Prompt forecasting method. For cost reasons, the forecaster-based filtering is only applied to the testing set, as context-aided probabilistic forecasting at this scale is prohibitively expensive with current methods. This yields unfiltered training data but rigorously validated testing data, which is arguably more critical for evaluation.

We partition the test samples by difficulty and encourage practitioners to explicitly mention which subset is used instead of randomly sampling from the full set. Windows that are too easy to forecast without context poorly measure a model's ability to exploit context, while extremely challenging windows would be unrepresentative of real-world use cases. To balance both extremes, we split the testing set according to Chronos's unimodal forecasting performance, using a Mean Absolute Scaled Error (MASE) threshold of 1.5. Windows with MASE above 1.5 form the HARD split; those below form the EASY split. We sample equally from each before context augmentation, obtaining 490 HARD and 414 EASY windows after filtering (904 total). See App. B for more details.

3 Experiments

We envision CAF-7M as primarily a training dataset, with its usefulness linked to whether it provides a good pretraining foundation for CAF models. To validate its utility, we ideally would train models on CAF-7M and evaluate them on diverse real-world benchmarks. However, underperformance on existing datasets leaves ambiguous whether models cannot use context or whether context lacks predictive value. We address this by evaluating on CAF-7M's verified test sets, where contexts are known to be useful. We also demonstrate that performance gains transfer to the real-world ChatTime benchmark (Wang et al., 2025) in App. G.1. We organize our analysis around two questions:



(a) CRPS. (b) Win Rate.

Figure 2: CRPS (↓) and Win Rate (↑) for Direct Prompt and ChatTime on the **HARD** split (Tab. 1), filtered into Accepted and Rejected Sets by GPT-5.2. We assess each method with correct context, no context, and swapped context from another window. Filtering is crucial: context in the Accepted (Rejected) set improves (worsens) performance. Context usefulness generalizes beyond GPT-5.2, with larger models benefiting more (Zhang et al., 2025). We use Qwen3-14B for DoubleCast (App. C.1) to balance cost and performance.

Table 1: Benchmarking on CAF-7M test set using CRPS and Win Rate for HARD, EASY, and ALL splits. Top: zero-shot methods. Bottom: methods trained on CAF-7M. "No context" and "swapped" show evaluation without context or with swapped context. **Best results**, second best results.

MODEL	ALL CRPS	ALL WIN	HARD CRPS	HARD WIN	EASY CRPS	EASY WIN
Chronos	0.278 ± 0.002	N/A	0.304 ± 0.002	N/A	0.247 ± 0.004	N/A
TimeLLM (ETTm1, 192)	0.635 ± 0.001	0.08	0.549 ± 0.001	0.12	0.738 ± 0.001	0.03
TimeLLM (ETTm1, 336)	3.309 ± 0.003	0.00	3.212 ± 0.004	0.00	3.410 ± 0.005	0.00
TimeLLM (ETTm2, 336)	0.803 ± 0.001	0.05	0.703 ± 0.001	0.07	0.921 ± 0.001	0.02
TimeLLM (ETTm2, 96)	1.406 ± 0.002	0.01	1.284 ± 0.002	0.02	1.546 ± 0.003	0.00
ChatTime	0.385 ± 0.001	0.25	0.407 ± 0.001	0.25	0.358 ± 0.002	0.25
DP GPT-5.2	**0.217 ± 0.001**	0.62	**0.212 ± 0.001**	0.75	0.223 ± 0.001	0.47
DP Qwen3-14B	0.296 ± 0.001	0.51	0.277 ± 0.002	0.63	0.319 ± 0.002	0.36
TimeLLM (CAF)	0.501 ± 0.001	0.17	0.428 ± 0.001	0.22	0.587 ± 0.001	0.10
TimeLLM (CAF) (no context)	-8.0%	-26.7%	-1.5%	-34.5%	-14.0%	-11.9%
TimeLLM (CAF) (swapped)	-1.1%	+8.7%	+2.0%	+7.3%	-3.6%	+7.1%
DoubleCast	0.231 ± 0.001	**0.71**	0.251 ± 0.001	**0.79**	**0.204 ± 0.002**	**0.64**
DoubleCast (no context)	+18.3%	-34.7%	+22.0%	-41.9%	+11.1%	-21.5%
DoubleCast (swapped)	+26.5%	-37.6%	+25.2%	-40.8%	+25.4%	-37.7%

RQ1: Does CAF-7M's testing set contain informative contexts, complementary to time series? We validate whether contexts in CAF-7M's test set contain complementary predictive signal by assessing whether they provide measurable forecasting improvements. We apply the Direct Prompt forecasting technique with a variety of generic LLMs, zero-shot, and with or without context (see App. F.4 for details on how Direct Prompt is used). We use the CRPS (Gneiting & Raftery, 2007) as our primary metric, normalized by mean absolute forecast ground truths to avoid large values dominating aggregated results, and the Win Rate (ratio of windows where Direct Prompt has lower CRPS than Chronos) as a complementary metric robust to outliers.

Figure 2 shows the impact on forecasting accuracy on the HARD split's accepted and rejected sets when context is withheld (No Context) or mismatched (Swapped Context). Contexts in the accepted set lead to significant improvements in both CRPS and Win Rate for Direct Prompt with any LLM backbone. Mismatched contexts do not improve performance over no context, confirming that performance gains are driven by complementary information in the contexts.

The importance of the filtering is revealed when comparing the Accepted Set and the Rejected Set in Fig. 2. In the Rejected Set, forecasts are worse with context than without. This highlights the need to filter synthetic contexts to avoid concluding that forecasting methods cannot leverage context when contexts are simply too low quality to make up a rigorous test set. Even though sample selection uses GPT-5.2, results are highly correlated across other models for both sets, suggesting accepted samples broadly contain useful predictive information that generalizes across forecasters.

RQ2: Does training on CAF-7M enable generalization to its testing set? To assess whether training on CAF-7M enables generalization to its verified test set, we train TimeLLM (Jin et al., 2024) and DoubleCast on it and compare against multiple zero-shot forecasting models: Chronos,

TimeLLM pre-trained on other datasets, ChatTime, and Direct Prompt using GPT-5.2 or Qwen3-14B (see App. F). See App. C for DoubleCast details.

Tab. 1 reports the different aggregated CRPS and Win Rates for these baselines and models trained on CAF-7M. DoubleCast achieves strong performance, second only to Direct Prompt with GPT-5.2, which was used to filter test samples. This indicates that while filtering is mandatory for good test sets, training on CAF-7M leads to performant CAF models, even with unfiltered training samples.

To check whether models trained on CAF-7M make use of the context, Tab. 1 also reports by how much the CRPS and Win Rate increase or decrease when the context is either omitted (`no context`) or randomly exchanged between windows (`swapped`). DoubleCast actively uses context since both metrics worsen markedly when context is missing or swapped, with CRPS increasing 11.1%-26.5% and Win Rate decreasing 21.5%-41.9%. This does not apply to TimeLLM trained on CAF-7M, as some results improve when context is removed or swapped, indicating TimeLLM was unable to properly use CAF-7M's context despite better results than on its original training data. While dataset scale is critical, model architecture plays an equally decisive role; further work is needed to understand why implicit alignment methods like Time-LLM fail to exploit high-quality context.

4 Conclusion

In this work, we presented **CAF-7M**, a 7-million-window context-aided forecasting (CAF) dataset of real-world time series augmented with diverse LLM-generated contexts and a test set of contexts verified for complementarity by state-of-the-art CAF methods. We also introduced DoubleCast, a new CAF architecture that, when trained on CAF-7M, performs competitively on our verified test split and generalizes to real-world CAF tasks. Our findings highlight that (i) measuring progress in CAF relies on benchmarks that explicitly validate context complementarity (ii) multi-modal CAF architectures can learn to leverage context, even from noisy synthetic datasets. Future work will focus on increasing context diversity, and reducing context complementarity verification costs.

References

Taha Aksu, Gerald Woo, Juncheng Liu, Xu Liu, Chenghao Liu, Silvio Savarese, Caiming Xiong, and Doyen Sahoo. GIFT-Eval: A benchmark for general time series forecasting model evaluation. In *NeurIPS Workshop on Time Series in the Age of Large Models*. 27

Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, November 2024. URL https://openreview.net/forum?id=gerNCVqqtR. 2, 3, 19, 24

Jialin Chen, Aosong Feng, Ziyu Zhao, Juan Garza, Gaukhar Nurbek, Cheng Qin, Ali Maatouk, Leandros Tassiulas, Yifeng Gao, and Rex Ying. MTBench: A multimodal time series benchmark for temporal reasoning and question answering. *arXiv preprint arXiv:2503.16858*, 2025. 9

Yunfeng Ge, Jiawei Li, Yiji Zhao, Haomin Wen, Zhao Li, Meikang Qiu, Hongyan Li, Ming Jin, and Shirui Pan. T2s: High-resolution time series generation with text-to-series diffusion models. *arXiv preprint arXiv:2505.02417*, 2025. 9

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 102(477):359–378, 2007. 3, 4

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024. 3

Nate Gruver, Marc Finzi, Shikai Qiu, and Andrew G Wilson. Large language models are zero-shot time series forecasters. *Advances in Neural Information Processing Systems*, 36:19622–19635, 2023. 10

Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice*. OTexts, 2018. 1

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-LLM: Time series forecasting by reprogramming large language models. In *Proceedings of the International Conference on Learning Representations*, 2024. URL https://arxiv.org/abs/2310.01728. 4, 10, 25

Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Kamarthi, Aditya B. Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, and B. Aditya Prakash. Time-MMD: A new multi-domain multimodal dataset for time series analysis, 2024a. 1, 9

Haoxin Liu, Shangqing Xu, Zhiyuan Zhao, Lingkai Kong, Harshavardhan Prabhakar Kamarthi, Aditya Sasanur, Megha Sharma, Jiaming Cui, Qingsong Wen, Chao Zhang, et al. Time-MMD: Multi-domain multimodal dataset for time series analysis. *Advances in Neural Information Processing Systems*, 37:77888–77933, 2024b. 2, 9

Xu Liu, Junfeng Hu, Yuan Li, Shizhe Diao, Yuxuan Liang, Bryan Hooi, and Roger Zimmermann. Unitime: A language-empowered unified model for cross-domain time series forecasting. In *Proceedings of the ACM Web Conference 2024*, pp. 4095–4106, 2024c. 10

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Autotimes: Autoregressive time series forecasters via large language models. *Advances in Neural Information Processing Systems*, 37:122154–122184, 2024d. 10

Mike A. Merrill, Mingtian Tan, Vinayak Gupta, Tom Hartvigsen, and Tim Althoff. Language models still struggle to zero-shot reason about time series, 2024. URL https://arxiv.org/abs/2404.11757. 3

OpenAI. GPT-5.2 API. https://platform.openai.com/, 2026. Accessed: 2026-01-27. 3

Zijie Pan, Yushan Jiang, Sahil Garg, Anderson Schneider, Yuriy Nevmyvaka, and Dongjin Song. s^2 IP-LLM: Semantic space informed prompt learning with LLM for time series forecasting. In *Forty-first International Conference on Machine Learning*, 2024. 10

James Requeima, John Bronskill, Dami Choi, Richard Turner, and David K Duvenaud. LLM processes: Numerical predictive distributions conditioned on natural language. *Advances in Neural Information Processing Systems*, 37:109609–109671, 2025. 10

Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by Layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025. 20

Mingtian Tan, Mike Merrill, Vinayak Gupta, Tim Althoff, and Tom Hartvigsen. Are language models actually useful for time series forecasting? *Advances in Neural Information Processing Systems*, 37:60162–60191, 2024. 10

Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chattime: A unified multimodal time series foundation model bridging numerical and textual data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 12694–12702, 2025. 1, 2, 3, 9, 10, 25, 26

Xinlei Wang, Maike Feng, Jing Qiu, Jinjin Gu, and Junhua Zhao. From news to forecast: Integrating event analysis in llm-based time series forecasting with reflection. *Advances in Neural Information Processing Systems*, 37:58118–58153, 2024. 2, 10

Andrew Robert Williams, Arjun Ashok, Étienne Marcotte, Valentina Zantedeschi, Jithendaraa Subramanian, Roland Riachi, James Requeima, Alexandre Lacoste, Irina Rish, Nicolas Chapados, and Alexandre Drouin. Context is Key: A benchmark for forecasting with essential textual information. In *Proceedings of the 2025 International Conference on Machine Learning (ICML 2025)*, May 2025. URL https://openreview.net/forum?id=ih2WuBT1Fn. Poster. 1, 3, 9, 10, 23, 25

Wenfa Wu, Guanyu Zhang, Zheng Tan, Yi Wang, and Hongsheng Qi. Dual-forecaster: A multimodal time series model integrating descriptive and predictive texts. *arXiv preprint arXiv:2505.01135*, 2025. 10

Wenyan Xu, Dawei Xiang, Yue Liu, Xiyu Wang, Yanxiang Ma, Liang Zhang, Chang Xu, and Jiaheng Zhang. FinMultiTime: A four-modal bilingual dataset for financial time-series analysis. *arXiv preprint arXiv:2506.05019*, 2025. 9

Zhijian Xu, Yuxuan Bian, Jianyuan Zhong, Xiangyu Wen, and Qiang Xu. Beyond trend and periodicity: Guiding time series forecasting with textual cues. *arXiv e-prints*, pp. arXiv–2405, 2024. 9

Hao Xue and Flora D Salim. Promptcast: A new prompt-based learning paradigm for time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 36(11):6851–6864, 2023. 10

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. 19

Xiyuan Zhang, Boran Han, Haoyang Fang, Abdul Fatir Ansari, Shuai Zhang, Danielle C Maddix, Cuixiong Hu, Andrew Gordon Wilson, Michael W Mahoney, Hao Wang, et al. Does multimodality lead to better time series forecasting? *arXiv preprint arXiv:2506.21611*, 2025. 1, 4, 9, 10, 19

Siru Zhong, Weilin Ruan, Ming Jin, Huan Li, Qingsong Wen, and Yuxuan Liang. Time-VLM: Exploring multimodal vision–language models for augmented time series forecasting. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, Proceedings of Machine Learning Research. PMLR, 2025. URL https://arxiv.org/abs/2502.04395. to appear. 10

Xin Zhou, Weiqing Wang, Francisco J Baldán, Wray Buntine, and Christoph Bergmeir. MoTime: A dataset suite for multimodal time series forecasting. *arXiv preprint arXiv:2505.15072*, 2025. 9

Appendix

CONTENTS

Table 2: Comparison of dataset size in terms of domains and textual contexts. We distinguish between independent texts and template-generated texts. Further, we emphasize datasets that are LLM-generated and provide only evaluation data. CAF-7M contains more than twice as many texts as any prior dataset while covering more diverse domains and providing both training and evaluation data.

Dataset	# Domains	# Ind. texts	# Templates	# Templated texts	LLM Generated	Test only
Time-MMD (Liu et al., 2024b)	9	26,880	–	–	✓	
FinMultiTime (Xu et al., 2025)	1	3,742,445*	–	–		
MoTime (Zhou et al., 2025)	4	910,908	–	–		
CGTSF (Wang et al., 2025)	3	–	3	33,693		
MTBench (Chen et al., 2025)	2	4,026*	–	–		
TGTSF (Xu et al., 2024)	2†	87,160‡	3	162*	✓	
TSFragment-600K (Ge et al., 2025)	5	700,800	–	–	✓	
Context is Key (CiK) (Williams et al., 2025)	8	–	71	355		✓
CAF-7M (Ours)	11	7,433,239	–	–	✓	

∗: These datasets contained many duplicated entries. These numbers only count non-duplicated texts.
†: Two of the datasets mentioned in the paper were not publicly available at time of writing.
‡: This number reflects the amount of LLM generated snippets which are then combined according to a template.

A Background and Related Work

A.1 Problem Setting

We consider the problem of *context-aided forecasting*, where given past observations $z_{1:P} \in \mathbb{R}^P$ of a univariate time series and arbitrary natural language context $\mathbf{x}$, the goal is to estimate a conditional predictive distribution over future values $z_{P+1:P+Q} \in \mathbb{R}^Q$,

$$p(z_{P+1:P+Q} \mid z_{1:P}, \mathbf{x}). \tag{1}$$

A context $\mathbf{x}$ is considered *useful* if conditioning on it improves forecast quality. Formally, let $\mathcal{L}$ be a proper scoring rule and let $z^\star := z^\star_{P+1:P+Q}$ denote a realization drawn from the true future distribution. Context $\mathbf{x}$ is useful if

$$\mathbb{E}_{z^\star}\big[\mathcal{L}(p(\cdot \mid z_{1:P}, \mathbf{x}), z^\star)\big] < \mathbb{E}_{z^\star}\big[\mathcal{L}(p(\cdot \mid z_{1:P}), z^\star)\big], \tag{2}$$

where lower values of $\mathcal{L}$ indicate better forecasts.

A.2 Multimodal Time Series Datasets

Several datasets for context-aided forecasting have been proposed, but ensuring context relevance remains challenging due to low signal-to-noise ratios in naturally available sources. One approach leverages domains where timestamped articles are naturally aligned with time series, as in FinMultiTime (Xu et al., 2025), MoTime (Zhou et al., 2025), and MTBench (Chen et al., 2025); to reduce input length, some use LLMs to summarize lengthy reports (Liu et al., 2024a). A second approach uses LLMs to generate descriptions from numerical patterns in the historical window, as in TGTSF (Xu et al., 2024) and TSFragment-600K (Ge et al., 2025). Finally, template-based methods convert metadata or summary statistics into text (Wang et al., 2025; Xu et al., 2024; Williams et al., 2025). Table 2 compares these datasets along key dimensions.

In line with previous work, our approach uses LLMs to generate scenarios aligned with time series windows, leveraging their broad background knowledge to produce diverse and plausible textual context. Unlike prior methods that condition only on historical values or restrict the impact of future values to rigid templates, we allow the LLM to be flexible in how it extracts information from both history and future values. This enables the LLM to generate context that is both genuinely informative for prediction while still being plausible. Furthermore, our approach enables filtering of generated context by comparing forecasting performance with and without it, allowing curation of data instances where context provides complementary predictive signal.

A.3 Models for Context-Aided Forecasting

Recent work in context-aided forecasting leverages language models to combine contextual and numerical data, following two paradigms (Zhang et al., 2025): (i) *alignment-based* methods, i.e.

multimodal architectures that fuse time-series representations with text embeddings (Jin et al., 2024; Liu et al., 2024c; Wang et al., 2025), or (ii) *prompt-based* methods, which are LLM-based forecasters that prompt large language models with numerical and textual inputs (Xue & Salim, 2023; Gruver et al., 2023; Requeima et al., 2025; Williams et al., 2025). A major drawback of prompting-based methods is their computational cost. For example, the Direct Prompt method from Williams et al. (2025) that we evaluate in Sec. 3 requires many tokens to generate a single floating-point number. In Appendix C.1, we introduce a distinct alignment-based method that leverages a specialized dual-attention decoder.

A.3.1 Alignment-Based Methods

Among the methods that align time series and textual modalities via dedicated architecture, UniTime (Liu et al., 2024c) concatenates time-series and textual embeddings, and feeds the resulting vector to a language-TS transformer based on GPT2. Time-LLM (Jin et al., 2024) introduces patch reprogramming to map time series into textual prompts and adds as prefix domain description, instructions and statistics of the time-series, leveraging frozen LLMs to perform forecasting with minimal training. Dual-forecaster (Wu et al., 2025) combines the time-series modality with textual descriptions of its history and its future via attention mechanisms. Time-VLM (Zhong et al., 2025) broadens the modality set by coupling vision-style encodings of temporal patterns with generated textual descriptions, then leveraging a frozen vision-language backbone to provide augmented representations that particularly help in few/zero-shot regimes.

A.3.2 Prompt-Based Methods

Another line of work prompt LLMs with both numerical and contextual inputs to generate probabilistic forecasts (Gruver et al., 2023; Requeima et al., 2025; Williams et al., 2025). Subsequent work tightens the alignment between language representations and time-series structure. S^2IP-LLM (Pan et al., 2024) learns prompts in a joint semantic space that aligns pre-trained LLM embeddings with time-series features, improving transfer while reducing heavy fine-tuning. In parallel, AutoTimes (Liu et al., 2024d) repurposes decoder-only LLMs as autoregressive forecasters by projecting time series into the language token space; Wang et al. (2024) use LLM agents to retrieve, filter, and align news events with time-series fluctuations, iteratively refining forecasts through event-aware reasoning. Tan et al. (2024) find that for several recent LLM-for-forecasting models, removing or replacing the LLM with simple attention often maintains or improves accuracy, suggesting that LLMs help most when they bring truly complementary knowledge, e.g., coming from textual sources. This question is further studied in Zhang et al. (2025), where the authors empirically identify the conditions by which textual context helps improve forecasting performance. In particular, they find that textual context is most beneficial when it conveys information not inferable from time series, such as domain metadata or future events, and that architectures which explicitly align the two modalities are more reliable than treating series purely as text.

B CAF-7M Dataset Details

B.1 Construction Pipeline and Splits

We construct CAF-7M from the Chronos datasets collection[2] via a staged pipeline: (i) extract supervised forecasting windows from each source dataset, (ii) attach auxiliary fields (baseline difficulty proxies and textual context), and (iii) materialize temporally consistent in-domain splits together with fully held-out zero-shot test sets.

Forecasting windows. For each dataset, we generate rolling forecasting windows consisting of a history $\mathbf{z}_{1:P}$ and a future horizon $\mathbf{z}_{P+1:P+Q}$. History length P is sampled uniformly from $[Q, 512]$, matching the maximum context length used by Chronos. The prediction length Q is determined by dataset frequency metadata: 64 for hourly, daily, business-daily series, 5-minutely, and 15-minutely; 48 for 30-minutely; 52 for weekly; 12 for monthly; 8 for quarterly; and 5 for yearly.

Window augmentation. Each window is augmented with: (i) a baseline forecast and difficulty proxy (e.g., Chronos MASE) used for *hard/easy* stratification; and (ii) an LLM-generated scenario-

[2] https://huggingface.co/datasets/autogluon/chronos_datasets

Table 3: Train and Test split datasets used for CAF-7M.

train	**test**
electricity_15min	dominick
m4_daily	ercot
m4_hourly	exchange_rate
m4_weekly	m4_quarterly
m4_monthly	m5
mexico_city_bikes	monash_car_parts
monash_electricity_hourly	monash_australian_electricity
monash_electricity_weekly	monash_m1_yearly
monash_kdd_cup_2018	monash_hospital
monash_london_smart_meters	monash_tourism_yearly
monash_pedestrian_counts	monash_tourism_quarterly
monash_rideshare	monash_tourism_monthly
monash_saugeenday	monash_m3_yearly
monash_temperature_rain	monash_cif_2016
solar	monash_m1_monthly
solar_1h	m4_yearly
taxi_1h	monash_covid_deaths
taxi_30min	monash_m1_quarterly
uber_tlc_daily	monash_m3_monthly
uber_tlc_hourly	monash_weather
ushcn_daily	monash_m3_quarterly
weatherbench_daily	monash_fred_md
weatherbench_hourly_10m_u_component_of_wind	monash_nn5_weekly
weatherbench_hourly_10m_v_component_of_wind	monash_traffic
weatherbench_hourly_2m_temperature	nn5
weatherbench_hourly_geopotential	
weatherbench_hourly_potential_vorticity	
weatherbench_hourly_relative_humidity	
weatherbench_hourly_specific_humidity	
weatherbench_hourly_temperature	
weatherbench_hourly_toa_incident_solar_radiation	
weatherbench_hourly_total_cloud_cover	
weatherbench_hourly_total_precipitation	
weatherbench_hourly_u_component_of_wind	
weatherbench_hourly_v_component_of_wind	
weatherbench_hourly_vorticity	
weatherbench_weekly	
wiki_daily_100k	
wind_farms_daily	
wind_farms_hourly	

style context. Context is produced by prompting `Llama-3-70B-Instruct` with dataset metadata and a serialized (history, future) pair using the template in Fig. 3. We use temperature $= 0.6$ and top_p $= 0.9$, following the recommended settings in the Hugging Face model card.[3] For scalability, we run the INT4-quantized checkpoint.[4]

Dataset partitioning. We partition the 65 source datasets into a training split of 40 datasets and a held-out test split of 25 datasets (see Table 3), following Chronos's *in-domain* and *zero-shot* evaluation protocol: the in-domain group is used for training, while the zero-shot group is excluded entirely from training and used only for evaluation. For in-domain datasets only, we compute dataset-specific temporal cutoffs from the empirical distribution of `start_idx` to avoid leakage. Specifically, for each in-domain dataset, we define a training cutoff at the 90th percentile and a validation cutoff at the 95th percentile of `start_idx`. These cutoffs are computed once and reused across all split generation steps. Using the cutoffs above, each in-domain dataset is split as:

- `indomain-train`: `start_idx` $\leq$ 90th percentile,
- `indomain-val`: 90th $<$ `start_idx` $\leq$ 95th percentile,
- `indomain-test`: `start_idx` $>$ 95th percentile.

We use `indomain-val` for training monitoring and `indomain-test` to evaluate in-domain fit; however, we make performance claims for DoubleCast only based on benchmarks on the held-out zero-shot datasets. We report DoubleCast results on `indomain-test` (HARD), `indomain-test` (EASY), and `indomain-test` (ALL). We define HARD and EASY using the Chronos MASE difficulty proxy: HARD if Chronos MASE > 1.5 and EASY otherwise, while ALL is the union of HARD and EASY.

For each zero-shot dataset, we construct a test split only. Because we do not train on these datasets, we do not enforce temporal cutoffs. Instead, we apply the same difficulty filtering described below to the full candidate pool, shuffle with a fixed seed, and sample to a fixed test budget. We report `zeroshot-test` (HARD), `zeroshot-test` (EASY), and `zeroshot-test` (ALL) (the union of HARD and EASY).

The primary splits of CAF-7M are `indomain-train`, which we called the training split of CAF-7M in the main text, and `zeroshot-test` (ALL), which we called the testing split of CAF-7M in the main text.

DP-based context verification for evaluation splits. To ensure that textual context provides *measurable* predictive utility, we construct a *verified* pool on evaluation splits via DP verification. For each candidate window, we query a downstream probabilistic forecaster twice—once with the generated context and once without—compute CRPS for both, and retain the window if the context improves CRPS. When drawing from this verified pool, we enforce the DP constraint alongside any temporal and/or difficulty constraints above. All evaluation splits in this work are produced with DP verification enabled. For `indomain-val` and `indomain-test`, we use `Qwen2.5-7B-Instruct` as the DP verifier to reduce cost. For `zeroshot-test`, we use `GPT-5.2` to maximize verification fidelity, yielding a higher-quality test set for assessing context complementarity.

Semantic validity pre-filter (zero-shot only). Because DP verification with `GPT-5.2` is expensive, we apply a lightweight semantic validity check before DP verification on zero-shot candidates using an LLM judge. Given a generated context, the judge answers two binary questions: (Q1) whether the context states what variable the time series represents, and (Q2) whether the described scenario/event is plausibly related to that variable (Fig. 4). This judge stage is used only as a coarse pre-filter to remove clearly invalid contexts; DP verification remains the primary acceptance mechanism.

Global splits and sampling. After generating per-dataset splits, we concatenate windows across all in-domain datasets to form global `indomain-train`, `indomain-val`, and `indomain-test` splits, and concatenate all zero-shot datasets to form a global `zeroshot-test` split. When reporting stratified evaluation variants (HARD/EASY/ALL), we sample windows from the corresponding candidate pools using a fixed seed and a fixed per-dataset or global budget.

[3] https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct

[4] https://huggingface.co/RedHatAI/Llama-3.3-70B-Instruct-quantized.w4a16

Training subsampling for scaling studies. Finally, we create subsampled variants of the in-domain training set (e.g., 0.1%, 1%, 10%, 25%, 50%, etc.) by shuffling training indices with a fixed seed and selecting the first n windows.

B.2 Dataset Statistics

In this subsection, we summarize the composition and difficulty characteristics of the CAF-7M splits through dataset-level statistics. Specifically, we report per-dataset window counts, the distributions of history and forecast-horizon lengths (both overall and stratified by dataset), and the baseline difficulty proxy Chronos MASE stratified by dataset. Dataset count bar charts sort datasets by window count in descending order. Length distributions are visualized using histograms and box-and-whisker plots. Boxplots follow Matplotlib defaults: boxes show the interquartile range with the median indicated, whiskers extend to $1.5 \times \mathrm{IQR}$, and outliers are not shown. For all MASE visualizations, we exclude non-finite values and values above 5 prior to plotting.

```
"
You are working with the following dataset:

{domain_hint}

Your task is to generate textual context that aids in forecasting the future of a time series
     from this dataset. The generated context should introduce information that helps explain
     the observed changes but is not directly obvious from the past numerical values. Below
     are examples of the style of contexts you should generate:

1. Background: This dataset represents electricity consumption, measured in kilowatts (kW), in
      City A. Scenario: Suppose a heat wave occurs in City A from 2013-05-28 12:00:00 to
     2013-05-28 14:00:00, resulting in excessive use of air conditioning and electricity
     consumption increasing to four times the usual level.
2. Constraint: In the forecast, the values are assumed to be bounded above by 6.29.

These examples illustrate the three key types of contextual information:

**Background**: Historical details that may influence future values.
**Scenario**: Plausible events that could lead to changes in future values.
**Constraint**: Assumptions or bounds that restrict the possible range of future values.

Your generated context should include one or more of these types. Now, using the following
     time series {target_var_text}from this dataset, generate a plausible textual context to
     help forecast. The data is in (timestamp, value) format, with the forecast horizon
     starting at {window['future_timestamp'][0]}.

Historical Time Series:
<past_target>
{history}
</past_target>

Ground Truth for Future:
<future_target>
{future}
</future_target>

Follow these guidelines when generating the context:

- **Focus** on non-stationary segments in the forecast horizon---such as trend shifts (e.g.,
     from fluctuating to a steady decline) or non-recurring patterns (e.g., unusually low
     peaks). A change is major if it shows a sustained, non-seasonal deviation (e.g., surge,
     drop, plateau) lasting at least one full seasonal cycle. Ignore minor fluctuations
     lasting only one or two timestamps.

- **If no major changes are present**, generate only a **Background** summary describing
     historical trends and seasonality, and how they are expected to persist.

- **If major changes are observed** (in all or part of the forecast):
- If multiple such changes occur, describe **only the most impactful one**, based on magnitude
      or duration.
- Create a realistic and specific **Scenario** describing a plausible causal event. If several
      plausible causes exist, choose the most likely one. Avoid vague or speculative reasoning
     .
- Include a **Constraint** only if forecast values are clearly and consistently bounded (e.g.,
      all values equal to 6.29). Omit this part entirely if no constraint is observed. Do not
     write "Constraint: None" or similar.
- Ensure the scenario is consistent with both the **domain description** and the patterns in <
     future_target>.
- Describe the impact on future values using **relative terms** (e.g., "2×" higher", "20%
     lower") rather than exact values.
- If the scenario affects only part of the forecast, clearly specify the **start and end times
     ** using the format YYYY-MM-DD HH:MM:SS.
Use HH:MM:SS only if all future timestamps are on the same day.
- You may refer to **historical timestamps** only if they clearly relate to recurring or
     causal patterns.
- Include a **short summary** in <short_desc> (e.g., "traffic jam", "holiday closure").
If no scenario is generated, return: <short_desc></short_desc>.

{variety_instruction}

- **Keep the context concise** --- ideally **150-250 words**. Avoid unnecessary elaboration or
      repetition.

- **Do not include your reasoning steps** or explain how the context was inferred. Output only
      the final description in a **factual and declarative** style.

Output Format:

<context_abs>put context here</context_abs>
<short_desc>put short description here if applicable</short_desc>
```

Figure 3: Template of the prompt sent to `Llama-3-70B-Instruct` for the context generation. {`domain_hint`} is filled with the dataset metadata; {`history`} and {`future`} are the serialized time series data; and {`variety_instruction`} contains a resume of the previously-generated context with instructions to not repeat it.

```
 "
Given the following context information about a time series:

{context}

Answer these two questions with ONLY "yes" or "no":

Q1: Does the context state what variable the time series represents (e.g., profit, demand,
     load), even if units or aggregation level are not specified?
Q2: Is the scenario/event related to the quantity being measured in the time series (i.e., it
     describes events that would reasonably affect that specific variable)?

Format your answer as:
Q1: [yes/no]
Q2: [yes/no]
```

Figure 4: Template of the prompt sent to `GPT-5.2` to judge the semantic validity of a window context. {`context`} is filled with the generated context.

`indomain-train`.

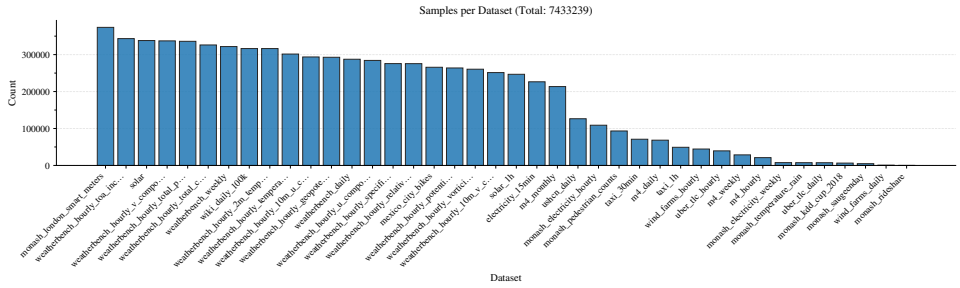


Figure 5: Per-dataset window counts for `indomain-train`.

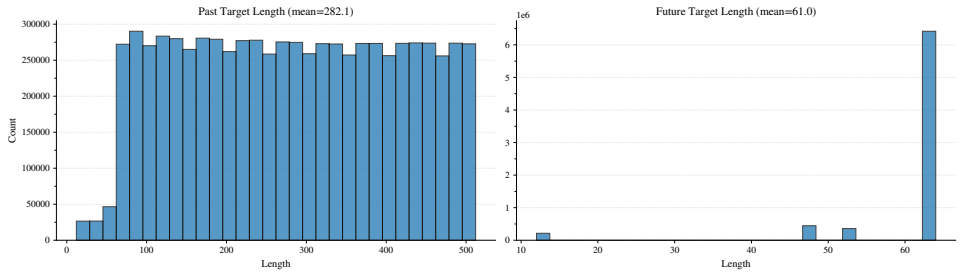


Figure 6: Overall distribution of history and forecast-horizon lengths for `indomain-train`.

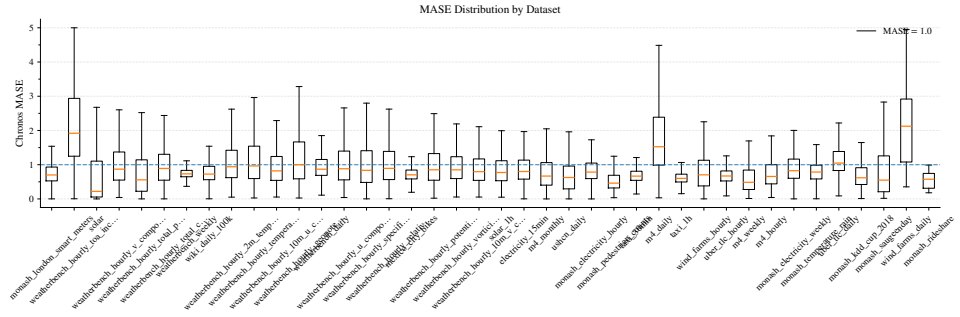


Figure 7: Baseline difficulty proxy (Chronos MASE) by dataset for `indomain-train`.

`indomain-test` (ALL).

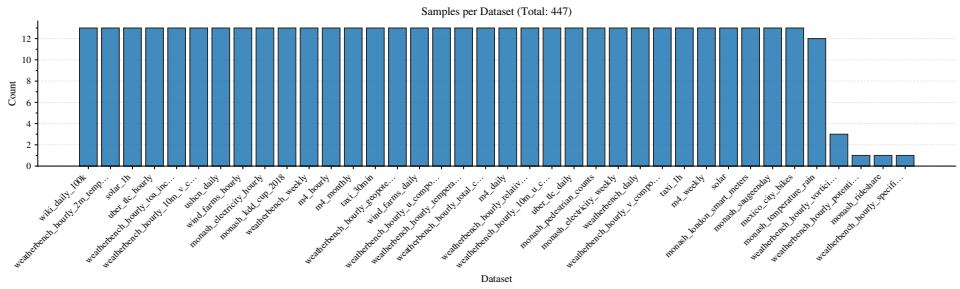


Figure 8: Per-dataset window counts for `indomain-test` (ALL).

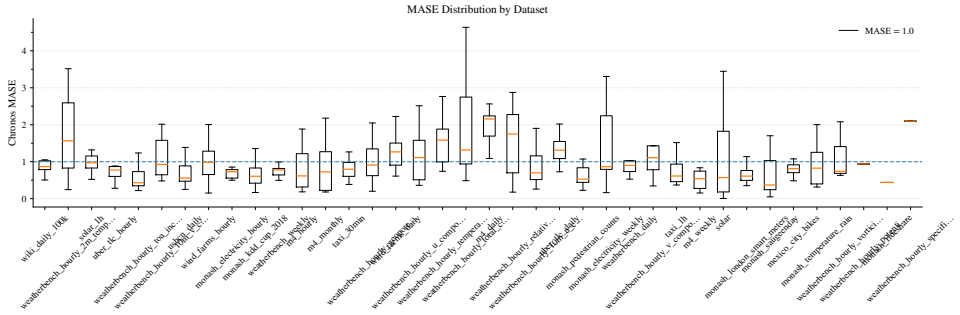


Figure 9: Baseline difficulty proxy (Chronos MASE) by dataset for `indomain-test` (ALL).

`indomain-test` (HARD).

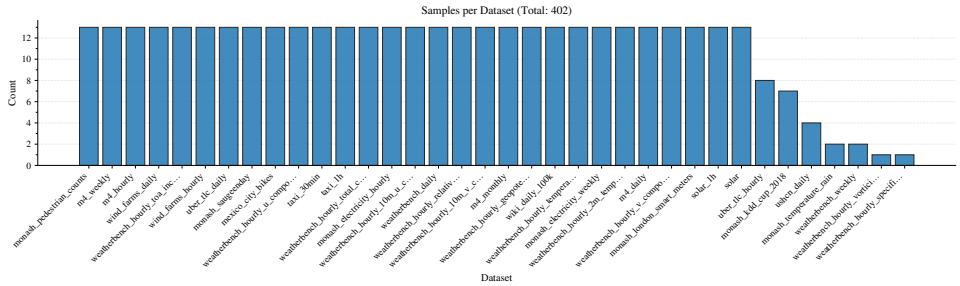


Figure 10: Per-dataset window counts for `indomain-test` (HARD).

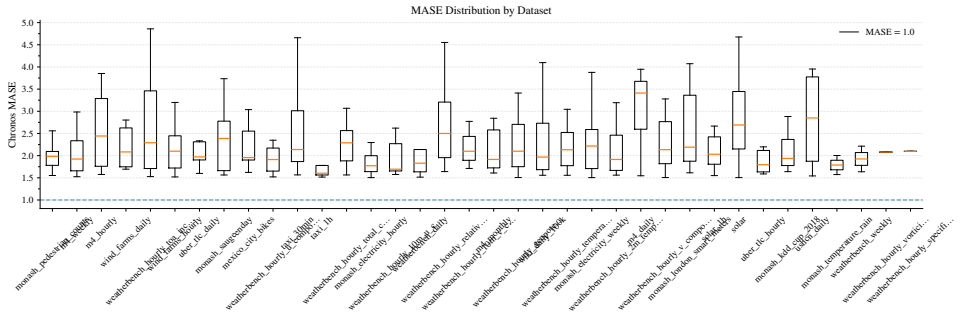


Figure 11: Baseline difficulty proxy (Chronos MASE) by dataset for `indomain-test` (HARD).

`indomain-test` (EASY).

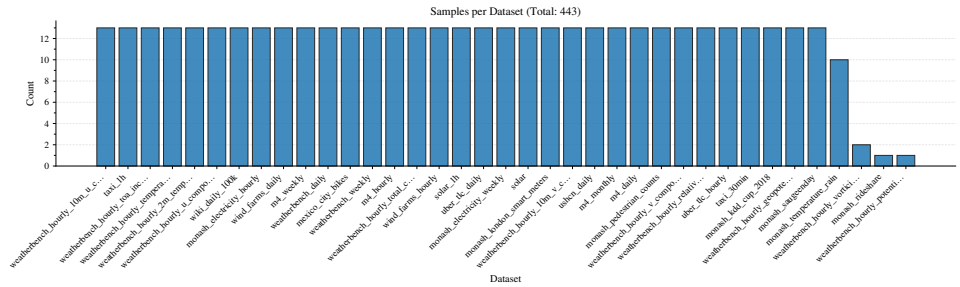


Figure 12: Per-dataset window counts for `indomain-test` (EASY).

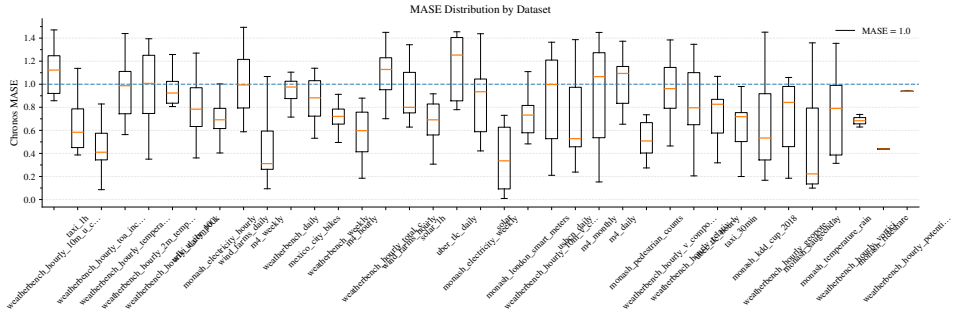


Figure 13: Baseline difficulty proxy (Chronos MASE) by dataset for `indomain-test` (EASY).

`zeroshot-test` (HARD).

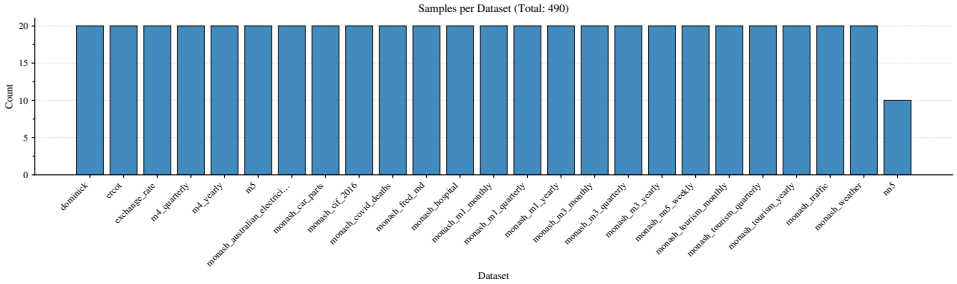


Figure 14: Per-dataset window counts for `zeroshot-test` (HARD).

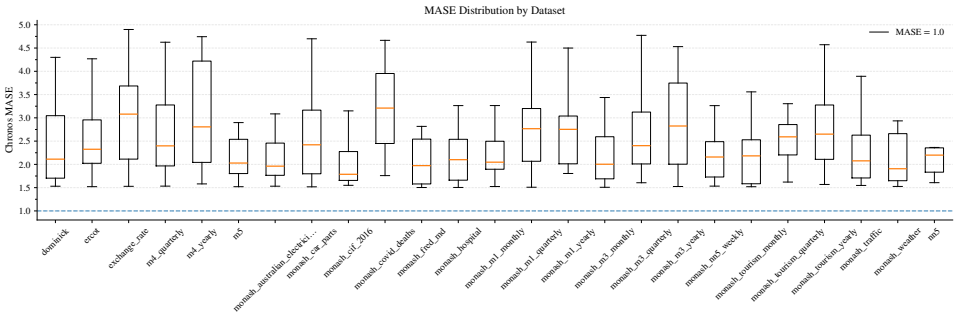


Figure 15: Baseline difficulty proxy (Chronos MASE) by dataset for `zeroshot-test` (HARD).

`zeroshot-test` (EASY).

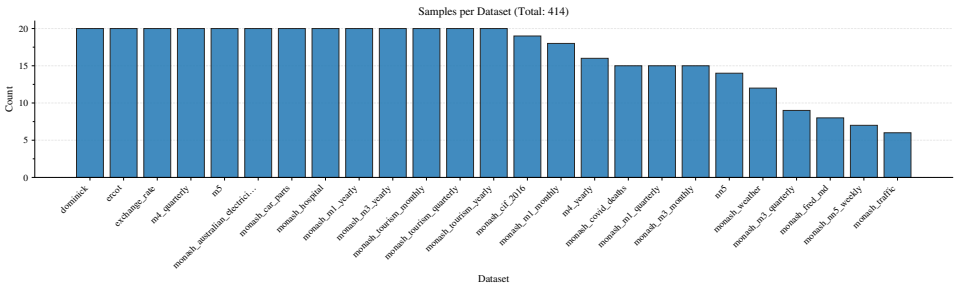


Figure 16: Per-dataset window counts for `zeroshot-test` (EASY).

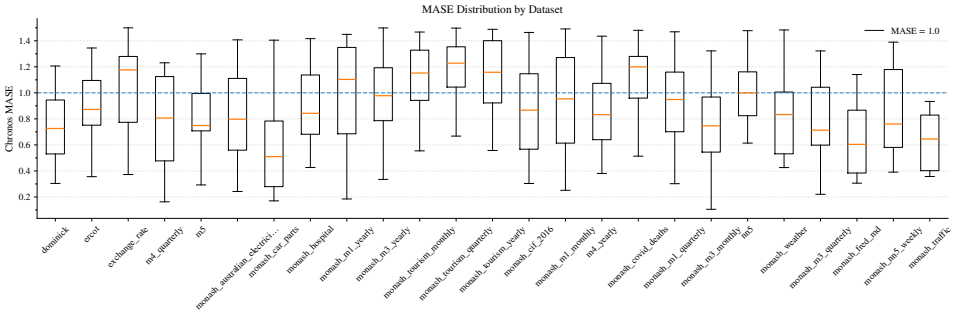


Figure 17: Baseline difficulty proxy (Chronos MASE) by dataset for `zeroshot-test` (EASY).

C DoubleCast Architecture

C.1 Training a Context-Aided Forecasting Model

To validate that CAF-7M can serve as an effective context-aided forecasting training dataset, we develop an alignment-based method for context-aided forecasting Zhang et al. (2025).

Aligning Context and Time Series There are many ways to combine the two modalities architecturally, such as concatenation, addition, or attention, and each with different capacity and computational trade-offs. A simple alignment-based fusion is as follows. A time-series encoder $\phi_z : \mathbb{R}^P \to \mathbb{R}^h$ maps the history $z_{1:P}$ to an h-vector $\mathbf{e}_{ts}$. A text encoder $\phi_x : \mathcal{X} \to \mathbb{R}^{h_x}$ embeds the unstructured context $\mathbf{x}$ into $\mathbf{e}_{ctx}$. We then fuse by concatenation via $\psi : \mathbb{R}^h \times \mathbb{R}^{h_x} \to \mathbb{R}^{h+h_x}$. Finally, a forecaster $g : \mathbb{R}^{h+h_x} \to \mathcal{P}(\mathbb{R}^Q)$ outputs the joint distribution over future values:

$$p\big(z_{P+1:P+Q} \mid z_{1:P}, \mathbf{x}\big) \;\approx\; g\big(\psi\big(\phi_z(z_{1:P}),\, \phi_x(\mathbf{x})\big)\big). \tag{3}$$

By pretraining on large corpora of contextual scenarios generated by LLMs, this approach learns to align textual context with temporal dynamics, producing forecasts that adapt dynamically to the conditions encoded in $\mathbf{x}$.

DoubleCast We introduce an alignment-based method which we call *DoubleCast*. DoubleCast synergizes the Chronos time series foundation model (Ansari et al., 2024) with an independent pre-trained text encoder to condition forecasts on unstructured text. The architecture consists of three primary components:

1. A **Time-Series Encoder**, which processes the historical time-series into a sequence of embeddings, $\boldsymbol{e}_{\text{ts}}$. This component is inherited directly from Chronos or another Time Series Foundation Model.
2. A **Text Encoder**: a pre-trained LLM (e.g., Qwen (Yang et al., 2025)) that encodes the context into a sequence of high-dimensional hidden states $\boldsymbol{e}_{\text{ctx}}$, which may be extracted from any chosen hidden layer.
3. A **DualT5 Decoder**, a decoder architecture which autoregressively generates the forecast by attending to both the time-series and text embeddings.

DoubleCast distinguishes itself by its specialized decoder: each Chronos decoder block is augmented with an additional *DualT5 cross attention layer* placed above the original Chronos encoder–decoder attention. This secondary cross-attention mechanism enables the decoder to integrate signals from the text encoder to condition on natural language context. Fig. 18 illustrates the whole architecture.

The central claims of our paper focus on the role of context in context-aided forecasting, not on the architecture. To keep the experimental focus on the data, we adopt a single, principled architectural configuration. Specifically, we use `Qwen3-14B` as the text encoder to balance quality and efficiency, extract embeddings from the second-to-last layer, and apply text-conditioning in every decoder block.

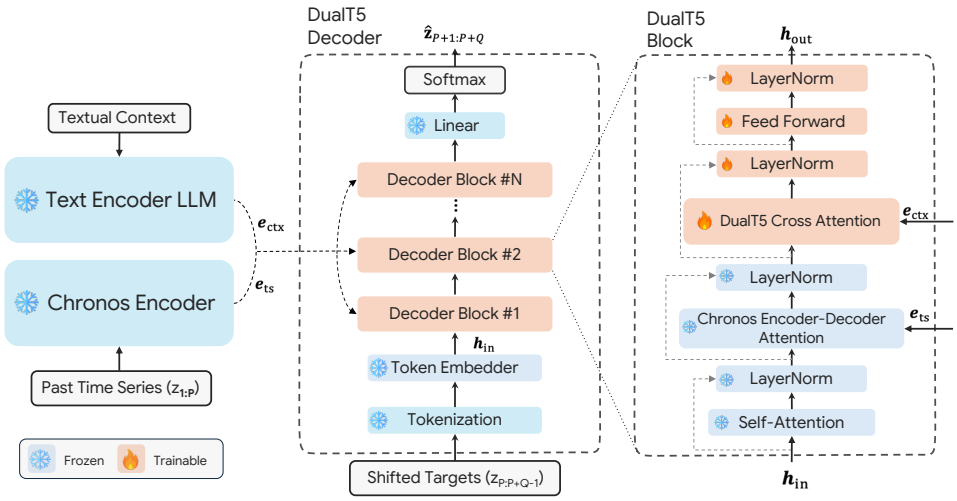


Figure 18: **Architecture of DoubleCast**. Each DualT5 decoder block consists of, in sequence: masked self-attention; Chronos encoder–decoder cross-attention; DualT5 cross-attention; and a FFN layer. Each sublayer is wrapped by a residual connection and layer normalization (LN). The same $\boldsymbol{e}_{\text{ts}}$ and $\boldsymbol{e}_{\text{ctx}}$ are provided to every decoder block.

C.2 DualT5 Decoder and Context Injection

DoubleCast augments the Chronos (T5) decoder with an additional text encoder–decoder cross-attention sublayer, enabling autoregressive forecasting conditioned on both numerical history and unstructured textual context. For a given forecasting window, the Chronos time-series encoder maps the historical series into encoder states $\boldsymbol{e}_{\text{ts}} \in \mathbb{R}^{L_{\text{ts}} \times d_{\text{model}}}$. In parallel, a pretrained text encoder maps the input context string into hidden states $\boldsymbol{e}_{\text{ctx}} \in \mathbb{R}^{L_{\text{ctx}} \times d_{\text{ctx}}}$. These same $\boldsymbol{e}_{\text{ts}}$ and $\boldsymbol{e}_{\text{ctx}}$ are provided to every (dual) decoder block, as shown in Fig. 18.

Block structure and ordering. A `DualT5` decoder block consists of the following sublayers in sequence: (i) masked self-attention, (ii) standard Chronos encoder–decoder cross-attention layer, over $\boldsymbol{e}_{\text{ts}}$, (iii) DualT5 cross-attention layer, over $\boldsymbol{e}_{\text{ctx}}$, and (iv) a feed-forward network (FFN). Each sublayer uses the standard T5 pre-norm convention: layer normalization is applied to the sublayer input, the transformation is applied, and the result is added back via a residual connection.

More concretely, within each decoder block, the intermediate hidden state $\tilde{\boldsymbol{h}}^{(0)}$ produced by the self-attention layer serves as the query for cross-attention over the time-series encoder outputs $\boldsymbol{e}_{\text{ts}}$, yielding $\tilde{\boldsymbol{h}}^{(1)}$. This state then becomes the query for a secondary cross-attention mechanism over the projected textual context embeddings $\boldsymbol{e}_{\text{ctx}}$. These embeddings are drawn from an intermediate layer of the pretrained LLM architecture, since these often capture rich semantic representations useful for downstream tasks (Skean et al., 2025).

To accommodate differing hidden dimensions between the text encoder and Chronos decoder, we implement cross-attention via an adapter layer: keys and values are generated by learned projections $W_K, W_V \in \mathbb{R}^{d_{\text{ctx}} \times d_{\text{ts}}}$, which map the text-encoder outputs (dimension d_{ctx}) into the decoder's attention space (dimension d_{ts}). This adapter-style design enables seamless integration of arbitrary LLMs without altering their native architectures.

Let $\boldsymbol{h}_{\text{in}}$ denote the input hidden state to a decoder block. The sequential updates can be written as

$$\tilde{\boldsymbol{h}}^{(0)} = \boldsymbol{h}_{\text{in}} + \text{SelfAttn}\big(\text{LN}(\boldsymbol{h}_{\text{in}})\big), \tag{4}$$

$$\tilde{\boldsymbol{h}}^{(1)} = \tilde{\boldsymbol{h}}^{(0)} + \text{CrossAttn}_{\text{ts}}\big(\text{LN}(\tilde{\boldsymbol{h}}^{(0)}),\, \boldsymbol{e}_{\text{ts}}\big), \tag{5}$$

$$\tilde{\boldsymbol{h}}^{(2)} = \tilde{\boldsymbol{h}}^{(1)} + \text{CrossAttn}_{\text{ctx}}\big(\text{LN}(\tilde{\boldsymbol{h}}^{(1)}),\, \boldsymbol{e}_{\text{ctx}}\big), \tag{6}$$

$$\boldsymbol{h}_{\text{out}} = \tilde{\boldsymbol{h}}^{(2)} + \text{FFN}\big(\text{LN}(\tilde{\boldsymbol{h}}^{(2)})\big), \tag{7}$$

where $\boldsymbol{h}_{\text{in}}$ denotes the input hidden state to the decoder block (either the output of the preceding block or the initial token embeddings); $\tilde{\boldsymbol{h}}^{(i)}$ for $i = 0, 1, 2$ are the intermediate hidden states; and $\boldsymbol{h}_{\text{out}}$ is the final hidden state of the decoder block.

where $\mathrm{CrossAttn}_{\text{ts}}$ is the original Chronos encoder–decoder attention and $\mathrm{CrossAttn}_{\text{ctx}}$ is the added text-conditioning attention.

Dual block placement. DoubleCast supports applying text-conditioning in all decoder layers or restricting it to a specified subset of decoder layers. Layers not selected remain standard T5 blocks and thus do not attend to $\boldsymbol{e}_{\text{ctx}}$. This provides a simple architectural knob to modulate where and how strongly text influences generation.

C.3 Text Encoder and Adapter Projections

Text encoder and feature selection. DoubleCast employs an independent pretrained text encoder to compute contextual embeddings. The model extracts textual representations from a configurable hidden layer index `text_encoder_layer_index` (default: second-to-last, -2), yielding $\boldsymbol{e}_{\text{ctx}} \in \mathbb{R}^{L_{\text{ctx}} \times d_{\text{ctx}}}$.

Adapter-style key/value projections. To accommodate dimensional mismatch between the text encoder width d_{ctx} and the Chronos decoder attention space, DoubleCast implements the text cross-attention via adapter projections on keys and values. Concretely, the added text attention reuses the standard T5 query projection from decoder states, but replaces the key/value projections with learned linear maps

$$\boldsymbol{K}_{\text{ctx}} = \boldsymbol{e}_{\text{ctx}} W_K, \qquad \boldsymbol{V}_{\text{ctx}} = \boldsymbol{e}_{\text{ctx}} W_V, \tag{8}$$

where $W_K, W_V \in \mathbb{R}^{d_{\text{ctx}} \times d_{\text{attn}}}$ map into the decoder attention inner dimension $d_{\text{attn}} = h \cdot d_{kv}$. This design enables seamless integration of arbitrary pretrained text encoders without modifying their native architectures.

Initialization and parameter-efficient tuning. The overall model is initialized from a pretrained Chronos checkpoint: all compatible encoder/decoder weights are copied, and only the newly introduced text cross-attention parameters are randomly initialized following T5 initialization. In our training setup, we freeze the entire backbone by default and selectively unfreeze (i) the text cross-attention parameters (including its layer norm) and (ii) the FFN sublayer within the dual blocks, yielding a parameter-efficient adaptation focused on learning how to condition Chronos forecasts on text.

C.4 Design Variants

Architectural variants. DoubleCast admits natural variants along three axes: (i) the choice of text encoder, (ii) the depth from which $\boldsymbol{e}_{\text{ctx}}$ is extracted (`text_encoder_layer_index`), and (iii) the set of decoder layers equipped with text cross-attention (`dual_block_placement`). In this work, we fix these degrees of freedom (Qwen3-14B, second-to-last layer, and text-conditioning in all decoder layers) to focus ablations on context quality and dataset scale rather than exhaustive architectural search.

D Limitations

Verifier bias and diversity limits at scale. Our methodology pairs TS windows with LLM-generated textual scenarios, and constructs a curated evaluation split using an LLM verifier: a window is *accepted* if adding the generated context reduces the verifier's CRPS, and *rejected* otherwise. This procedure yields a benchmark with measurable context complementarity, but can bias the accepted subset toward context patterns that the specific verifier can exploit, while potentially under-representing windows where context is helpful in ways not captured by the verifier. Moreover, verifier-based filtering is expensive and therefore not applied to the 7M-window training corpus, leaving training contexts potentially noisy or weakly aligned. Finally, generating textual scenarios at massive scale can exhibit limited diversity (e.g., repetitive phrasing or overuse of common event templates) despite explicit diversity prompting, which reduces the effective information content of the contextual modality and may encourage shallow keyword-to-pattern correlations. While our

generalization results (App. G.1) show that CAF-7M displays simulated-to-real transfer, these issues could be addressed by running efficient verifiers, such as strong alignment-based forecasters, as well as improved generation strategies (e.g., mixture-of-prompts, retrieval augmentation, or explicit diversity constraints).

DoubleCast's failure modes. DoubleCast conditions a Chronos-style decoder on text via additional cross-attention layers with learned projections. This is lightweight and effective, but it does not explicitly enforce fine-grained, time-local grounding between specific phrases and forecast steps. As a result, the model can fail in cases where the context specifies *precise temporal localization* (e.g., "a disruption occurs during the first two days of the horizon" or "a rebound begins around week 6") and accurate forecasting requires mapping that description to an exact region of the prediction window. Without an explicit mechanism to bind textual time references to forecast indices (e.g., event-time tags, time-aligned token routing, or a structured event representation), cross-attention may diffuse the signal across the horizon, leading to smeared effects (e.g., anticipating the change too early, delaying it, or spreading it over too many steps). This limitation is amplified when contexts contain multiple time-scoped events or when the horizon is long relative to the stated time frame, making it challenging to "locate" the relevant context region reliably.

E Training and Evaluation Protocols

E.1 Training and Inference Settings

Implementation and reproducibility. All experiments are implemented in `PyTorch` using `HuggingFace Transformers` training utilities. We fix the random seed to 42 for Python and NumPy/`torch` RNGs.

Model initialization. We initialize DoubleCast from a pretrained Chronos backbone (default: `amazon/chronos-t5-large`[5]) and a pretrained text encoder (e.g., `Qwen/Qwen3-14B`[6]). Text features are extracted from a specified hidden layer of the text encoder (default: second-to-last). The dual cross-attention modules are injected into a configurable subset of decoder blocks; unless otherwise noted, we enable them for all decoder blocks.

Trainable parameters. To stabilize optimization and isolate the effect of context fusion, we freeze all pretrained parameters and only fine-tune the newly introduced (or context-facing) components within each `DualT5DecoderBlock`. Concretely, for each enabled dual block we unfreeze (i) the cross-attention parameters (`EncDecAttention`), (ii) the corresponding layer-norm parameters, and (iii) the block feed-forward network (FFN). All other parameters remain frozen throughout training.

Optimization and training setup. We train with AdamW (`adamw_torch_fused`) using a cosine learning-rate schedule and weight decay. The global learning rate is 10^{-4} with warmup ratio 0.005 and weight decay 0.01, and we apply gradient clipping with max norm 1.0. Training runs for a fixed number of optimizer steps (`max_steps`=400,000) with gradient accumulation (`gradient_accumulation_steps`=2) and per-device batch size (`per_device_train_batch_size`=8). We save checkpoints every `save_steps` and evaluate every `eval_steps` on all validation datasets. Unless otherwise noted, we report results from the final checkpoint at 400,000 steps. To regularize the fusion pathway, we optionally down-weight updates to FFN parameters inside the dual blocks by applying a multiplicative learning-rate factor (default 0.01) via separate optimizer parameter groups. Training is conducted on a single NVIDIA H100 (80 GB) GPU using `bfloat16` weights and activations.

Fine-tuning on CGTSF (ChatTime). To adapt DoubleCast to CGTSF, we fine-tune the pretrained DoubleCast on the training split of CGTSF dataset. Using the checkpoint of DoubleCast trained in CAF-7M, we fine-tune for additional 50,000 steps using AdamW with a cosine schedule, learning rate 10^{-6}, warmup ratio 0.005, weight decay 0.01, and gradient clipping at max norm 1.0. We use per-device batch size 8 with gradient accumulation 2 (effective batch size 16). Similarly, during fine-tuning, we train only the text cross-attention, layer-norm, and FFN in each decoder block while

[5] https://huggingface.co/amazon/chronos-t5-large
[6] https://huggingface.co/Qwen/Qwen3-14B

keeping others frozen. In contrast to pretraining, we set the FFN learning-rate multiplier to 0.5 to allow more rapid adaptation of the decoder feed-forward sublayers to the target context style.

Prompt formatting for textual context. We serialize the textual context (and optional metadata such as forecast start date, frequency, and scale) into a single string that is tokenized by the text encoder. We consider two templates: a *naive* template that passes the raw context, and a *structured* template that wraps metadata and context in explicit tags, as illustrated in Fig. 19. Unless otherwise stated, DoubleCast uses the structured template in all experiments.

```
"
<info>
forecast_start_date=2025-06-25 00:00:00
frequency=D
scale_factor=0.1234
</info>

<context>
Background: The time series exhibits seasonal fluctuations...
</context>
```

Figure 19: Text encoder prompt.

Inference and test evaluation. At inference time, each evaluation instance provides a numerical history `past_target` and a textual context string. The forecast horizon T is set to the instance-specific future length, i.e., $T = |\texttt{future\_target}|$.

DoubleCast produces probabilistic forecasts via ancestral sampling from the decoder with `num_samples` Monte Carlo trajectories. We use the same generation procedure as Chronos—sampling with temperature and nucleus/top-k filtering as specified by the Chronos configuration—and generate sequences in chunks when T exceeds the model's native prediction length. Concretely, the pipeline iteratively predicts up to `prediction_length` steps per call and, when additional steps are required, appends the per-step sample median to the numerical context before continuing, ensuring a consistent autoregressive rollout while retaining stochasticity within each chunk. For comparability and efficiency, we run inference in `bfloat16` on GPU

We report CRPS (and standard error across windows) using the same empirical-sample estimator as in Appendix E.2. Finally, when Chronos reference forecasts are available for the same split, we compute the per-window CRPS win-rate of DoubleCast against Chronos under identical evaluation conditions.

E.2 Metrics

Notation. For an evaluation window with horizon length T, let the ground truth be $y_{1:T} \in \mathbb{R}^T$ and the probabilistic forecast be represented by N Monte Carlo samples $\{x_{1:T}^{(n)}\}_{n=1}^{N}$.

CRPS. We use the Continuous Ranked Probability Score (CRPS) as the primary probabilistic metric, following the standard empirical-sample formulation used in Williams et al. (2025). For each horizon step t, CRPS can be written as

$$\mathrm{CRPS}(X_t, y_t) = \mathbb{E}|X_t - y_t| - \tfrac{1}{2}\mathbb{E}|X_t - X_t'|, \tag{9}$$

where X_t and X_t' are i.i.d. draws from the empirical distribution defined by the N samples at step t. We report the horizon-averaged CRPS

$$\mathrm{CRPS}_{\mathrm{mean}}(X, y) = \frac{1}{T}\sum_{t=1}^{T} \mathrm{CRPS}(X_t, y_t). \tag{10}$$

To compare across heterogeneous series scales, we normalize by the mean absolute target magnitude

$$s(y) = \frac{1}{T}\sum_{t=1}^{T} |y_t|, \qquad \mathrm{nCRPS}(X, y) = \frac{\mathrm{CRPS}_{\mathrm{mean}}(X, y)}{s(y) + \varepsilon}, \tag{11}$$

with $\varepsilon = 10^{-6}$ for numerical stability. As in our evaluation pipeline, we cap per-window scores at $c = 5$:

$$\widetilde{\mathrm{nCRPS}}(X, y) = \min\{\mathrm{nCRPS}(X, y), c\}. \tag{12}$$

We then aggregate over a split by the simple mean (all instance weights are 1):

$$\overline{\mathrm{nCRPS}} = \frac{1}{M}\sum_{i=1}^{M} \widetilde{\mathrm{nCRPS}}_i. \tag{13}$$

Win-rate. Win-rate is computed against Chronos on a per-window basis using the same (normalized, clipped) CRPS. Let m_i be the model score and b_i the Chronos score on window i. The win-rate is

$$\mathrm{WinRate}(m \prec b) = \frac{1}{M}\sum_{i=1}^{M} \mathbb{I}[m_i < b_i], \tag{14}$$

where ties are counted in the denominator but not as wins.

MAE. For CGTSF we follow ChatTime and report normalized MAE. We form a point forecast via the per-timestep sample median,

$$\hat{x}_t = \mathrm{median}\{x_t^{(n)}\}_{n=1}^{N}, \tag{15}$$

and compute

$$\mathrm{MAE}(X, y) = \frac{1}{T}\sum_{t=1}^{T} |\hat{x}_t - y_t|, \qquad \mathrm{nMAE}(X, y) = \frac{\mathrm{MAE}(X, y)}{s(y) + \varepsilon}, \tag{16}$$

with the same scale $s(y)$ as in Eq. equation 11. We apply the same per-window cap $c = 5$ and report the mean over a split:

$$\overline{\mathrm{nMAE}} = \frac{1}{M}\sum_{i=1}^{M} \min\{\mathrm{nMAE}_i, c\}. \tag{17}$$

MASE. We also compute the Mean Absolute Scaled Error (MASE) to characterize per-window forecast difficulty. For a window with horizon length T, let $\hat{y}_{1:T}$ be the Chronos point forecast and $y_{1:T}$ the ground truth. Let $z_{1:L}$ denote the historical context used for conditioning. Given a seasonal period S, we define the scaling term

$$d(z; S) = \begin{cases} \frac{1}{L-S}\sum_{t=S+1}^{L} |z_t - z_{t-S}|, & L > S, \\ \frac{1}{L-1}\sum_{t=2}^{L} |z_t - z_{t-1}|, & L \le S, \end{cases} \tag{18}$$

i.e., the mean absolute error of a seasonal naive forecast on the history, with a non-seasonal fallback when insufficient history is available. The window-level MASE is then

$$\mathrm{MASE}(\hat{y}, y; z, S) = \begin{cases} \frac{\frac{1}{T}\sum_{t=1}^{T} |\hat{y}_t - y_t|}{d(z;S)}, & d(z; S) > 0, \\ 0, & d(z; S) = 0 \ \wedge \ \frac{1}{T}\sum_{t=1}^{T} |\hat{y}_t - y_t| = 0, \\ +\infty, & d(z; S) = 0 \ \wedge \ \frac{1}{T}\sum_{t=1}^{T} |\hat{y}_t - y_t| > 0. \end{cases} \tag{19}$$

We select S deterministically from the dataset frequency metadata (e.g., S=24 for hourly, S=7 for daily, S=12 for monthly; otherwise S=1).

F Baselines

F.1 Chronos

We use the publicly available implementation of Chronos (Ansari et al., 2024) from `https://github.com/amazon-science/chronos-forecasting`. In this work, we report results using `chronos-large` and run all Chronos inference on a single H100 GPU.

F.2 ChatTime

We evaluate ChatTime in the zero-shot setting using the released `ChatTime-1-7B-Chat` checkpoint (`https://huggingface.co/ChengsenWang/ChatTime-1-7B-Chat`), following the evaluation procedure provided in the authors' repository (`https://github.com/ForestsKing/ChatTime`).

F.3 TIME-LLM

We implement the Time-LLM (Jin et al., 2024) architecture using the authors' official codebase (https://github.com/KimMeen/Time-LLM). For zero-shot evaluation, we use the publicly released checkpoints and settings reported by Williams et al. (2025). For in-domain adaptation on CAF-7M, we further train Time-LLM on our `indomain-train` split. Concretely, we train for $400{,}000$ optimization steps with BF16 mixed precision on a single H100 GPU, using batch size 8 with gradient accumulation 2, AdamW with learning rate 10^{-4}, warmup ratio 0.005, weight decay 0.01, and gradient clipping at 1.0. We use a forecasting setup with sequence length 128, label length 64, and prediction length 64, and adopt the default long-term forecasting task configuration. Unless otherwise stated, we train a lightweight projection head with $d_{\text{model}} = 32$ and $d_{\text{ff}} = 128$, and optimize for MAE. Similarly, the final-checkpoint was used for experiments in this paper.

F.4 DIRECT PROMPT (DP)

Following prior work on prompt-based contextual forecasting (Williams et al., 2025), we evaluate DP baselines that queries an instruction-tuned LLM to produce the full forecast in a single pass. Given an instance-level textual context, a history window formatted as $(\text{timestamp}, \text{value})$ pairs, and a list of prediction timestamps, the model is prompted to output the forecast in the same $(\text{timestamp}, \text{value})$ format inside `<forecast>` tags. We use fixed prompt templates (with and without context) and refer to Figs. 20 and 21 for the exact prompts. Following the methodology in (Williams et al., 2025), we sample 25 independently generated forecasts per instance.

```
"
I have a time series forecasting task for you.

Here is some context about the task. Make sure to factor in any background knowledge,
satisfy any constraints, and respect any scenarios.
<context>
{context}
</context>

Here is a historical time series in (timestamp, value) format:
<history>
{past_time}
</history>

Now please predict the value at the following timestamps: {future_time_index_concat}.

Return the forecast in (timestamp, value) format in between <forecast> and </forecast> tags.
Do not include any other information (e.g., comments) in the forecast.

Example:
<history>
(t1, v1)
(t2, v2)
(t3, v3)
</history>
<forecast>
(t4, v4)
(t5, v5)
</forecast>
```

Figure 20: Template of the prompt for the Direct Prompt forecasting method. {`context`} is filled with the entry context; {`past_time`} is the serialized time series data from the history window; and {`future_time_index_concat`} contains the timestamps at which to do the forecast.

G ADDITIONAL RESULTS AND ANALYSES

G.1 GENERALIZATION TO REAL-WORLD BENCHMARKS

We finally assess whether a model trained on CAF-7M can generalize to other real-world datasets that do not share the same domains or structure. To that end, we use a test split from the CGTSF dataset Wang et al. (2025) as an out-of-domain benchmark for DoubleCast. We compare 4 models on the 3 CGTSF datasets: Chronos, as the unimodal version of DoubleCast; DoubleCast solely trained on

```
"
I have a time series forecasting task for you.

Here is a historical time series in (timestamp, value) format:
<history>
{past_time}
</history>

Now please predict the value at the following timestamps: {future_time_index_concat}.

Return the forecast in (timestamp, value) format in between <forecast> and </forecast> tags.
Do not include any other information (e.g., comments) in the forecast.

Example:
<history>
(t1, v1)
(t2, v2)
(t3, v3)
</history>
<forecast>
(t4, v4)
(t5, v5)
</forecast>
```

Figure 21: Template of the prompt for the Direct Prompt forecasting method when called without the context. {past_time} is the serialized time series data from the history window and {future_time_index_concat} contains the timestamps at which to do the forecast.

Table 4: CRPS (↓) on real-world CGTSF datasets, which have been used to train ChatTime Wang et al. (2025), for various history and prediction lengths. We show results for both DoubleCast trained solely on CAF-7M and DoubleCast pretrained on CAF-7M before being finetuned on the training split of CGTSF. The best and second best CRPS values for each dataset are bolded and underlined, respectively. DoubleCast without finetuning outperforms Chronos and ChatTime on LEU and PTF, but struggles with MSPG's prediction length of 96, which exceeds CAF-7M's maximum of 64 (indicated by the *). However, DoubleCast finetuned on CGTSF performs best, demonstrating that training on CAF-7M leads to context-processing capabilities that transfer to real-world datasets.

Dataset	Hist	Pred	Chronos	DoubleCast	DoubleCast (ft. on CGTSF)	ChatTime
PTF	48	24	0.1598 ± 0.0009	0.1534 ± 0.0007	**0.1517 ± 0.0008**	0.2073 ± 0.0004
	72	24	0.1401 ± 0.0008	0.1355 ± 0.0007	**0.1336 ± 0.0007**	0.1816 ± 0.0005
	96	24	0.1343 ± 0.0007	0.1294 ± 0.0007	**0.1279 ± 0.0007**	0.1693 ± 0.0005
	120	24	0.1297 ± 0.0007	0.1236 ± 0.0007	**0.1234 ± 0.0007**	0.1478 ± 0.0004
LEU	96	48	0.4646 ± 0.0012	0.4585 ± 0.0011	**0.4566 ± 0.0012**	0.4892 ± 0.0006
	144	48	0.4562 ± 0.0017	0.4497 ± 0.0012	**0.4492 ± 0.0011**	0.4826 ± 0.0006
	192	48	0.4504 ± 0.0014	0.4461 ± 0.0009	**0.4456 ± 0.0011**	0.4740 ± 0.0007
	240	48	0.4516 ± 0.0018	**0.4434 ± 0.0009**	0.4437 ± 0.0014	0.4640 ± 0.0006
MSPG	192	96*	**0.5568 ± 0.0026**	0.7683 ± 0.0024	0.6173 ± 0.0025	0.9746 ± 0.0010
	288	96*	0.4693 ± 0.0023	0.6054 ± 0.0025	**0.4649 ± 0.0023**	0.9679 ± 0.0010
	384	96*	0.4359 ± 0.0023	0.5675 ± 0.0025	**0.4169 ± 0.0021**	0.9661 ± 0.0011
	480	96*	0.4191 ± 0.0023	0.5160 ± 0.0024	**0.3891 ± 0.0022**	0.9655 ± 0.0011

CAF-7M, to determine its zero-shot capabilities; DoubleCast pre-trained on CAF-7M and finetuned on CGTSF, to see if it can serve as a good starting point for finetuning; and finally the official release of ChatTime, which was trained on CGTSF.

In Tab. 4, we observe that the finetuned version of DoubleCast has better performance than both Chronos and ChatTime on all but one of the dataset-prediction length pairs. This consistent advantage provides strong evidence that CAF-7M serves as an effective pretraining dataset with demonstrated simulated-to-real transfer.

The results for DoubleCast solely trained on CAF-7M are nuanced: while it is always more accurate than ChatTime, it is only better than Chronos on LEU and PTF, falling short on MSPG. Since the gap with the finetuned results is smaller than with the Chronos results on LEU and PTF, we conclude that DoubleCast trained on CAF-7M is able to zero-shot leverage the contextual data of these splits,

but that data has limited complementarity with the historical time series. As for MSPG, the small gap between Chronos and finetuned DoubleCast indicates that the contexts contain limited auxiliary information for the forecast.

In contrast to LEU and PTF, MSPG requires 96-step predictions, which exceeds CAF-7M's 64-step maximum. When Chronos and DoubleCast forecast on data with a prediction length greater than 64, the forecasting is done auto-regressively on chunks of 64 time steps. This means that the context is given to DoubleCast twice, while it was not trained it to apply the context on the second call (time steps 65 to 96). This distribution mismatch explains the poor zero-shot MSPG performance, making the finetuned model's success particularly striking: it learns to extend context application beyond its pretraining horizon.

Performance on Standard Numerical Forecasting Tasks A crucial concern for CAF-7M is whether training a model for contextual awareness degrades the model's numerical forecasting performance (i.e. in forecasting tasks with no context). To assess this, we evaluate DoubleCast on GIFT-Eval, a purely numerical time series forecasting benchmark (Aksu et al.) that is designed to measure general-purpose forecasting performance. Fig. 22 shows that DoubleCast retains Chronos' forecasting abilities: while performance degrades slightly, DoubleCast and Chronos perform almost identically when compared against AutoArima and Migas-1.0, the current leading statistical and pre-trained models on GIFT-Eval.

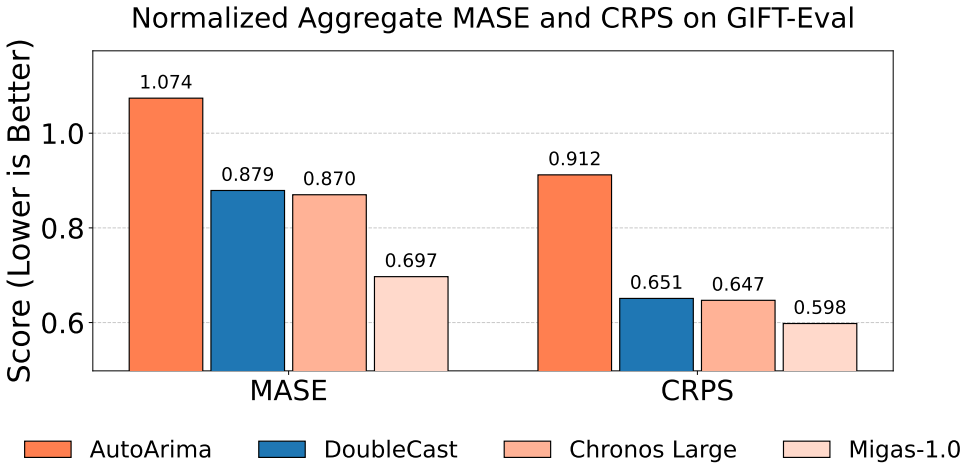
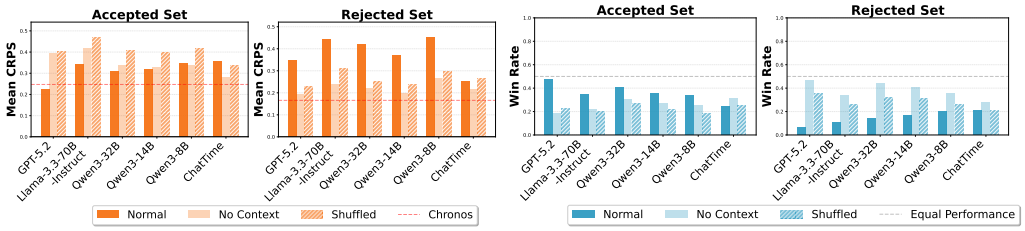


Figure 22: Normalized MASE (left, ↓) and CRPS (right, ↓) on GIFT-Eval (Aksu et al.). Despite extending Chronos for context-aided forecasting, DoubleCast retains Chronos' forecasting capabilities: DoubleCast performs nearly identically to Chronos on no-context time series forecasting, as compared to both AutoArima and the incumbent state-of-the-art pre-trained model (Migas-1.0).

G.2 Context Complementarity in the CAF-7M

Fig. 23 reports mean CRPS (left) and win-rate versus Chronos (right) on the EASY split, using the same accepted/rejected partition and the same three context settings (correct context, no context, swapped context). The overall trends match the HARD split but are generally attenuated, consistent with the fact that EASY windows already admit strong forecasts from numerical history alone. On the *accepted* set, providing the original (aligned) context reduces CRPS relative to the no-context baseline across most forecasters, and this improvement is reflected by higher win-rates. However, win-rates on EASY tend to remain closer to the equal-performance baseline, indicating smaller margins over Chronos even when context is informative. The *swapped-context* control again degrades performance, increasing CRPS and lowering win-rate compared to the aligned setting, which supports the interpretation that gains arise from instance-level context alignment rather than superficial properties of the text. Finally, on the *rejected* set, adding context typically worsens CRPS and reduces win-rate relative to no-context, as expected given that these windows are explicitly those where the verifier does not benefit from context under the acceptance criterion. Overall, the EASY results corroborate the benchmark construction: accepted windows exhibit measurable



(a) CRPS. (b) Win-rate.

Figure 23: Comparison of the CRPS and Win Rate for Direct Prompt using various LLMs and ChatTime. These results are only for windows where Chronos MASE is under 1.5 (EASY split). The results are split between windows which passed the filter described in Sec. 2 (Accepted Set) and those which did not (Rejected Set). Only windows from the Accepted Set are included in CAF-7M testing set.

context complementarity (albeit with less headroom than HARD), while rejected windows provide a counterfactual control where context is non-informative or misleading.

G.3 Scaling Laws: Training DoubleCast with Varying Data Fractions

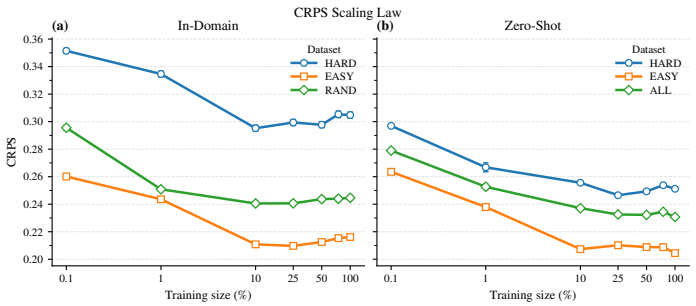


Figure 24: Effect of dataset scaling on CRPS performance.

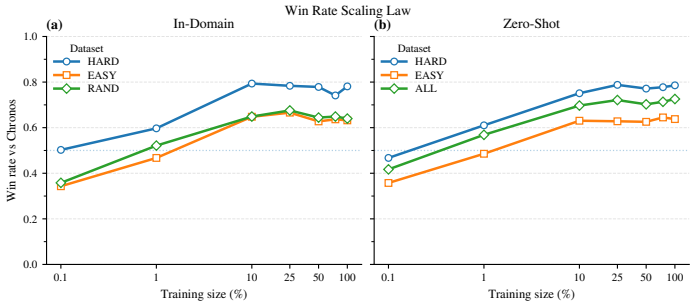


Figure 25: Effect of dataset scaling on win rate across models.

Figs. 24–25 report a data-scaling study where DoubleCast is trained with increasing fractions of `indomain-train` (x-axis: training size in %), and evaluated on both in-domain and zero-shot splits, stratified by difficulty. Overall, increasing the training fraction consistently improves performance: mean CRPS decreases as more training windows are used (Fig. 24), and win-rate versus Chronos increases in parallel (Fig. 25). The largest gains occur when moving from very small regimes (0.1%–1%) to moderate regimes (10%–25%), after which improvements tend to taper, indicating diminishing returns at larger data fractions. These trends are observed both in-domain and under distribution shift on the zero-shot benchmark.

G.4 DP-BASED CONTEXT VERIFICATION

G.4.1 VERIFICATION RESULTS

For both `zeroshot-test` (HARD) and `zeroshot-test` (EASY), we visualize the DP-based verification process using (i) *Acceptance funnel* plots (Figs. 26, 28) report how many candidate windows are (a) processed, (b) pass the lightweight semantic judge (`judge_passes` = Q1 & Q2), and (c) are ultimately accepted by DP verification; each stage is annotated with the corresponding percentage of the processed total. (ii) *Per-dataset breakdown* plots (Figs. 27, 29) aggregate outcomes by dataset and report: counts of judge-passed and DP-accepted windows; the judge pass rate (passed divided by total); and the "context helps" rate (accepted divided by passed).

`zeroshot-test` (HARD).

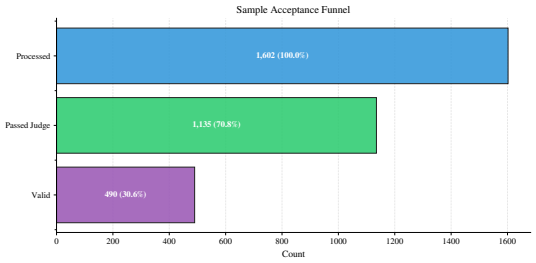


Figure 26: Acceptance funnel for DP-based verification on `zeroshot-test` (HARD).

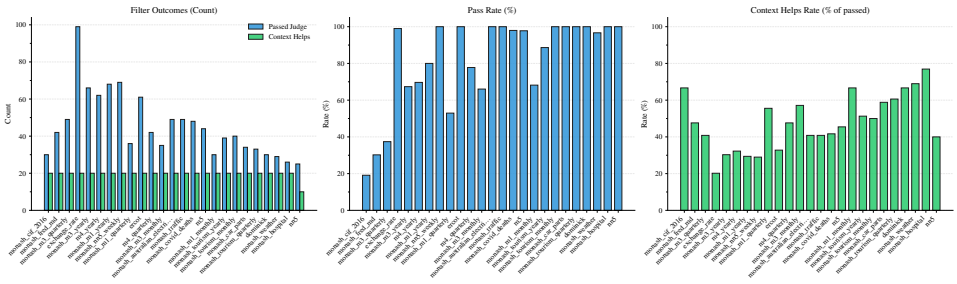


Figure 27: Per-dataset breakdown of accepted/rejected windows under DP-based verification on `zeroshot-test` (HARD).

`zeroshot-test` (EASY).

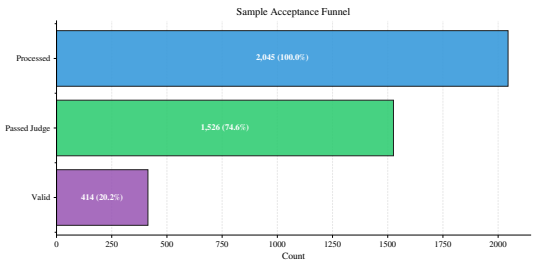


Figure 28: Acceptance funnel for DP-based verification on `zeroshot-test` (EASY).

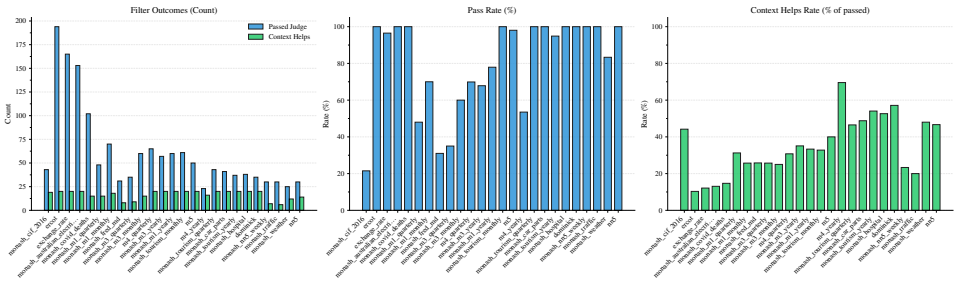


Figure 29: Per-dataset breakdown of accepted/rejected windows under DP-based verification on `zeroshot-test` (EASY).

G.4.2 Summary Statistics Before vs. After Filtering

This subsection tests whether DP-based verification (and associated filtering) induces unintended selection bias toward particular window characteristics. For each evaluation setting (`indomain-test`: ALL/HARD/EASY; `zeroshot-test`: HARD/EASY), we compare the *filtered* split against a *random* baseline of identical size, drawn uniformly from the same candidate pool prior to verification. Each figure overlays normalized histograms (density) of three window-level attributes: the baseline difficulty proxy, history length, and forecast-horizon length. Across both in-domain (Figs. 31–33) and zero-shot (Figs. 34, 35), the filtered distributions closely track their size-matched random counterparts, indicating that verification does not systematically skew the splits toward atypical history lengths or horizons. Deviations in the Chronos MASE panel are expected when conditioning on HARD/EASY, since these variants explicitly stratify windows by the difficulty proxy; importantly, this stratification does not propagate into shifts in history length or forecast length. Together, these checks support that the DP acceptance criterion primarily filters by the predictive usefulness of context rather than by trivial window geometry or sampling artifacts.

G.5 Impact of the Choice of LLM for Filtering

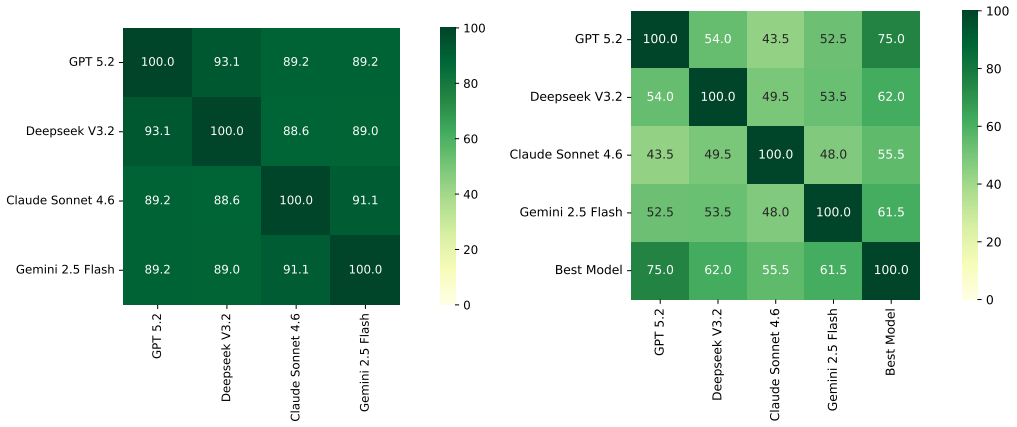
As described in App. B.1, the CAF-7M testing split was filtered using `GPT-5.2`. To determine how much of an impact on the final filtered dataset this choice of LLM has, we compare the results of this filtering step using `GPT 5.2` versus using 3 alternative LLMs: `Deepseek V3.2`, `Claude Sonnet 4.6`, and `Gemini 2.5 Flash`. Due to increased cost of the DP verification step, this experiment was done on a subset of 200 windows from the HARD set.

As can be seen in Fig. 30a, all 4 models have very high agreements about whether the windows pass the semantic judge filtering step. The lowest agreement with `GPT 5.2` are `Claude Sonnet 4.6` and `Gemini 2.5 Flash`, both with an agreement of 89.2%. This high agreement points that this step of the filtering process is mostly unaffected by the choice of LLM, as long as the LLM chosen is strong enough at the task.

However, Fig. 30b shows that the opposite is true for the DP verification step, with a agreement with `GPT 5.2` that goes from 43.5% to only 54.0%. This indicates that, unlike for the semantic judge step, the choice of LLM for the DP verification is crucial, since the set of windows which

Table 5: How often the forecasts produced using Direct Prompt with different models has the lowest CRPS values amongst their peers. These results are using 200 windows from a partially filtered version of CAF-7M HARD test set, after the semantic filtering step but prior to the DP verification step.

Model	Win rate without context (%)	Win rate with context (%)
`GPT 5.2`	23.5	**46.0**
`Deepseek V3.2`	27.5	18.5
`Claude Sonnet 4.6`	22.5	17.5
`Gemini 2.5 Flash`	25.5	18.0



(a) Agreement rates for the semantic judge.

(b) Agreement rates for the DP verification.

Figure 30: Comparison of how often various LLMs agreed about whether a window was considered valid by the semantic judge or by the DP verification steps. *Best Model* represents using the smallest CRPS value from all 4 models instead of using a single model.

passes this step would be completely different if another LLM was to be used instead of `GPT 5.2`. Nevertheless, this experiment shows that among those 4 models, `GPT 5.2` is indeed the LLM that is best suited for this task. Indeed, in idealized circumstances, this verification step would use the best possible context-aided forecasting method which can be approximated by the *Best Model* version, which uses the most accurate forecast (lowest CRPS) from the forecasts from the 4 models. Since `GPT 5.2` has the highest agreement (75.0%) with *Best Model*, it is thus the most appropriate model for the DP verification. To understand why `GPT 5.2` agree more often with *Best Model*, Tab. 5 shows how often its forecasts are the most accurate: while `GPT 5.2` is not superior to the other LLMs when forecasting without context, it is definitely superior on context-aided forecasting.

`indomain-test`.

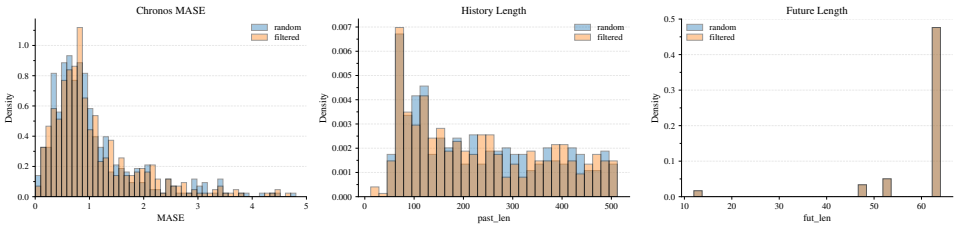


Figure 31: `indomain-test` (ALL): summary statistics before vs. after filtering.

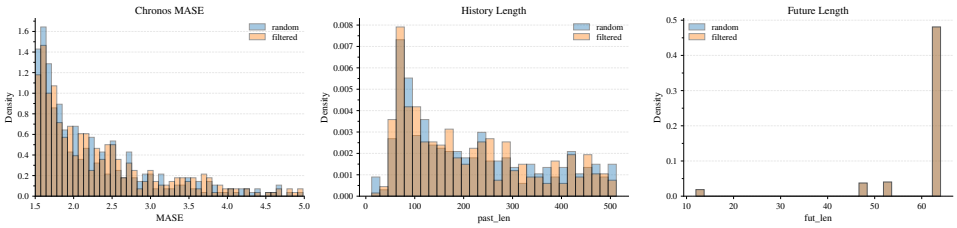


Figure 32: `indomain-test` (HARD): summary statistics before vs. after filtering.

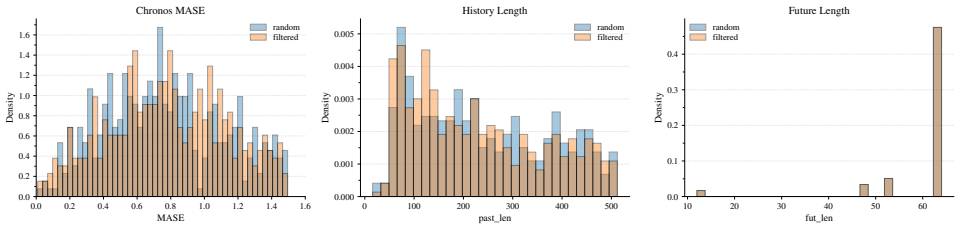


Figure 33: `indomain-test` (EASY): summary statistics before vs. after filtering.

`zeroshot-test`.

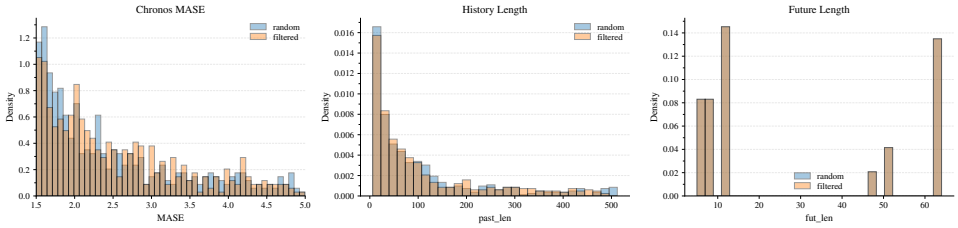


Figure 34: `zeroshot-test` (HARD): summary statistics before vs. after filtering.

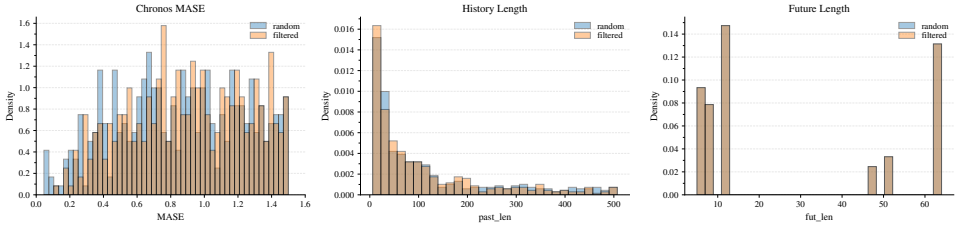


Figure 35: `zeroshot-test` (EASY): summary statistics before vs. after filtering.

G.6 QUALITATIVE FORECAST EXAMPLES: WHEN CONTEXT HELPS VS. HURTS

G.6.1 ZEROSHOT-TEST (HARD): CONTEXT HELPS

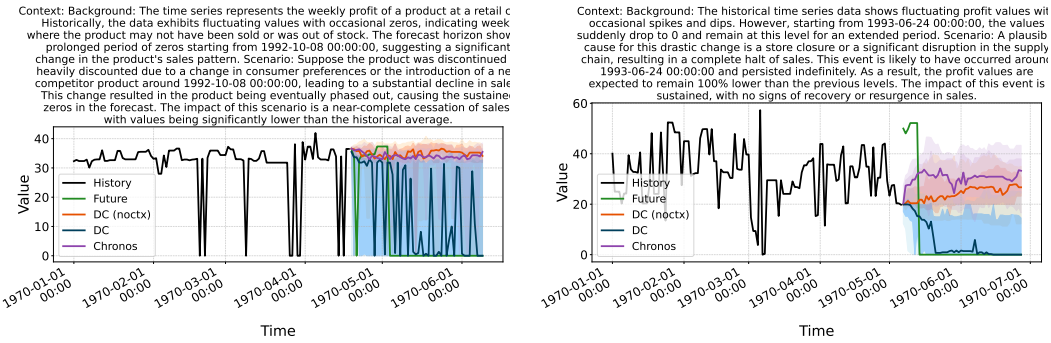


Figure 36: Examples of windows in `zeroshot-test` (HARD) where the context was found to be helpful when forecasting using DoubleCast.

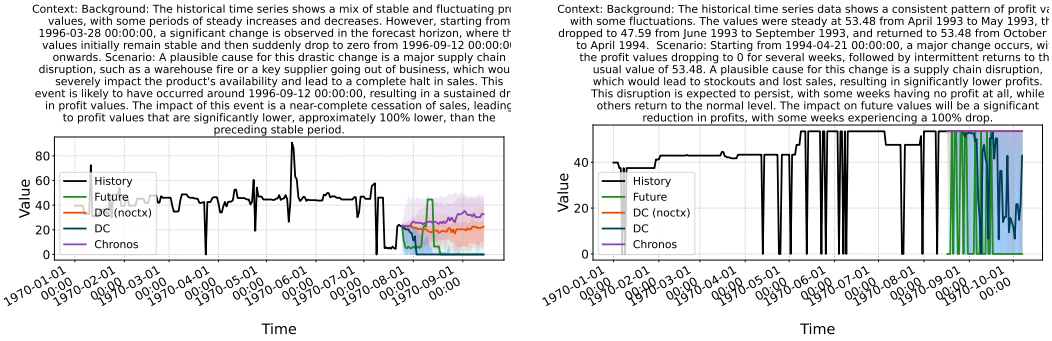


Figure 37: Examples of windows in `zeroshot-test` (HARD) where the context was found to be helpful when forecasting using DoubleCast.

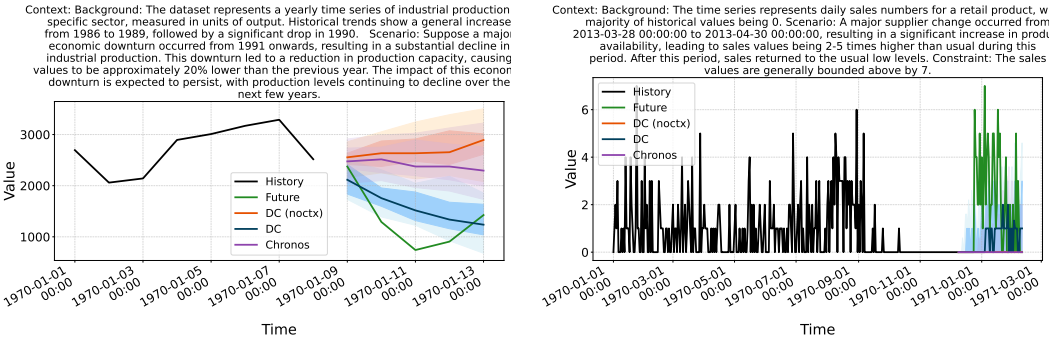


Figure 38: Examples of windows in `zeroshot-test` (HARD) where the context was found to be helpful when forecasting using DoubleCast.

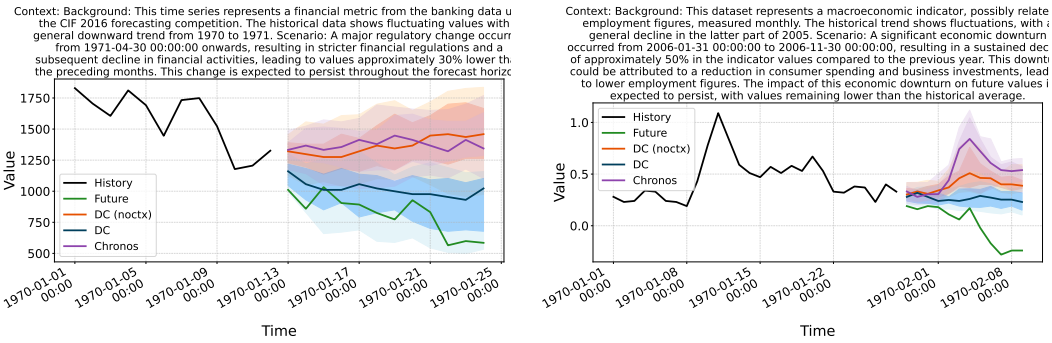


Figure 39: Examples of windows in `zeroshot-test` (HARD) where the context was found to be helpful when forecasting using DoubleCast.

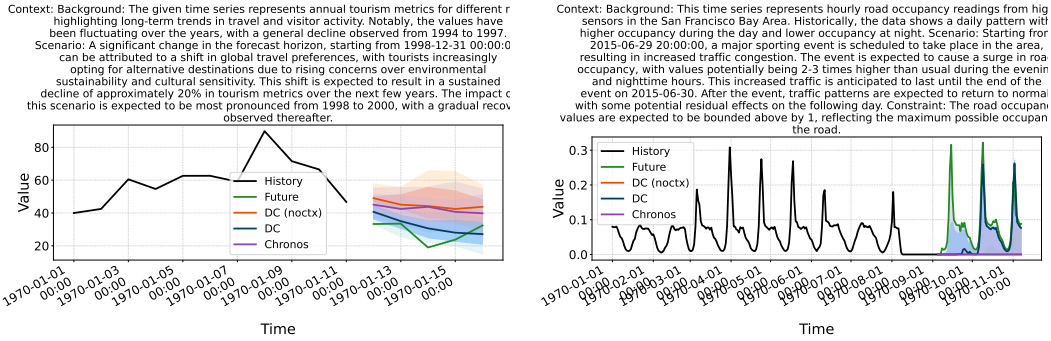


Figure 40: Examples of windows in `zeroshot-test` (HARD) where the context was found to be helpful when forecasting using DoubleCast.

G.6.2 ZEROSHOT-TEST (HARD): CONTEXT HURTS

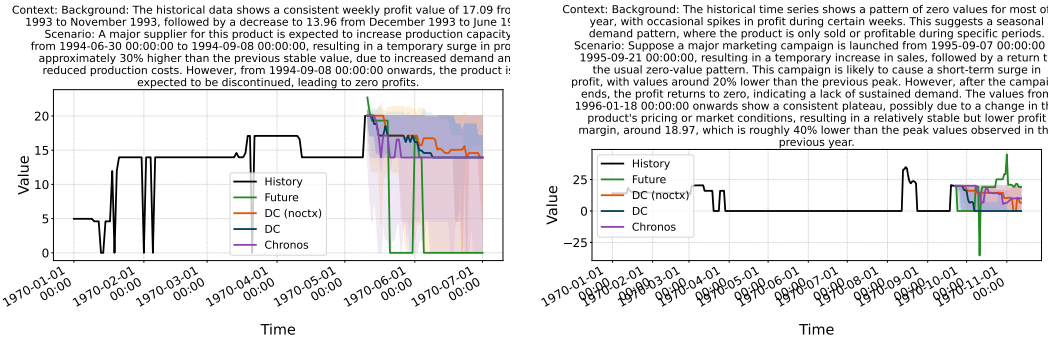


Figure 41: Examples of windows in `zeroshot-test` (HARD) where the context was found to not be helpful when forecasting using DoubleCast.

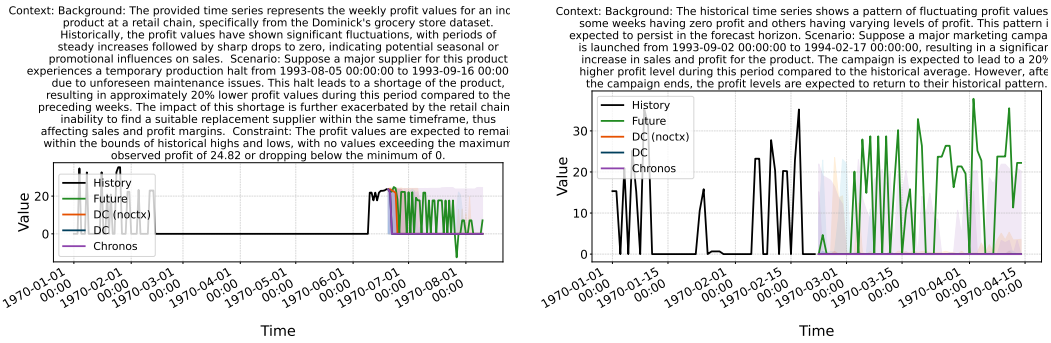


Figure 42: Examples of windows in `zeroshot-test` (HARD) where the context was found to not be helpful when forecasting using DoubleCast.

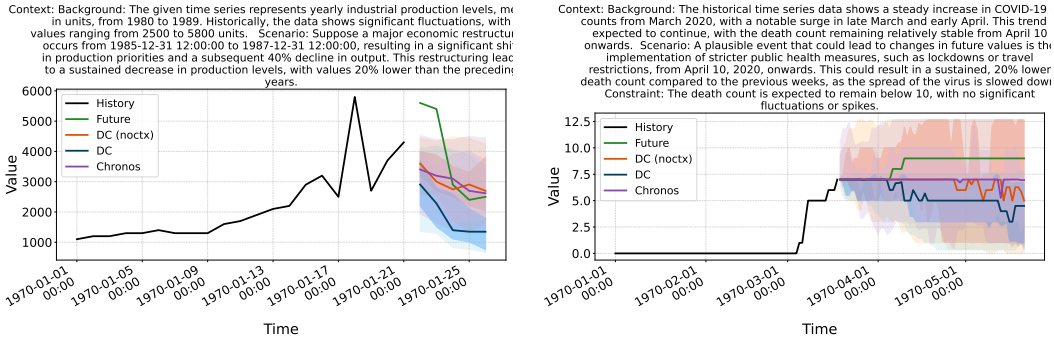


Figure 43: Examples of windows in `zeroshot-test` (HARD) where the context was found to not be helpful when forecasting using DoubleCast.

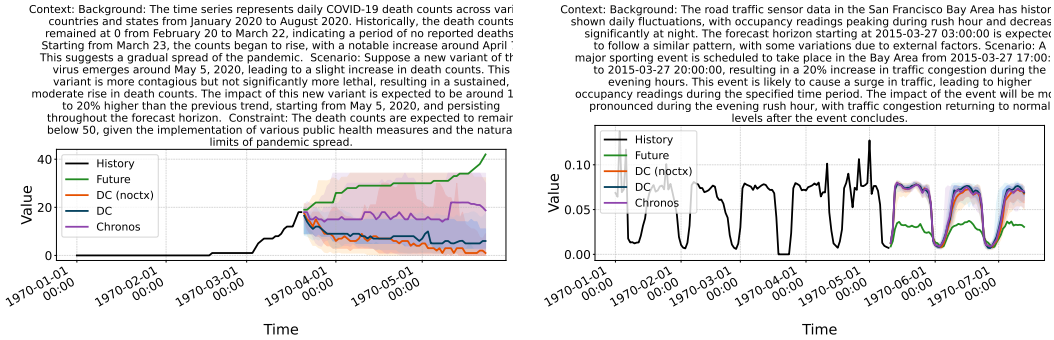


Figure 44: Examples of windows in `zeroshot-test` (HARD) where the context was found to not be helpful when forecasting using DoubleCast.

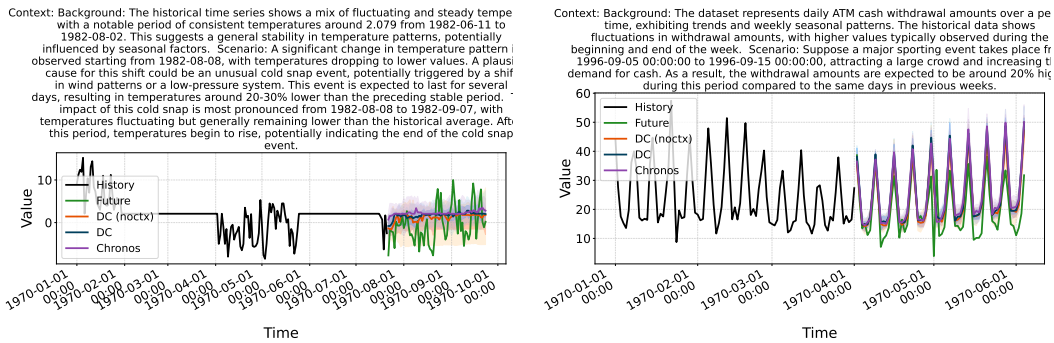


Figure 45: Examples of windows in `zeroshot-test` (HARD) where the context was found to not be helpful when forecasting using DoubleCast.

G.6.3 zeroshot-test (EASY): CONTEXT HELPS

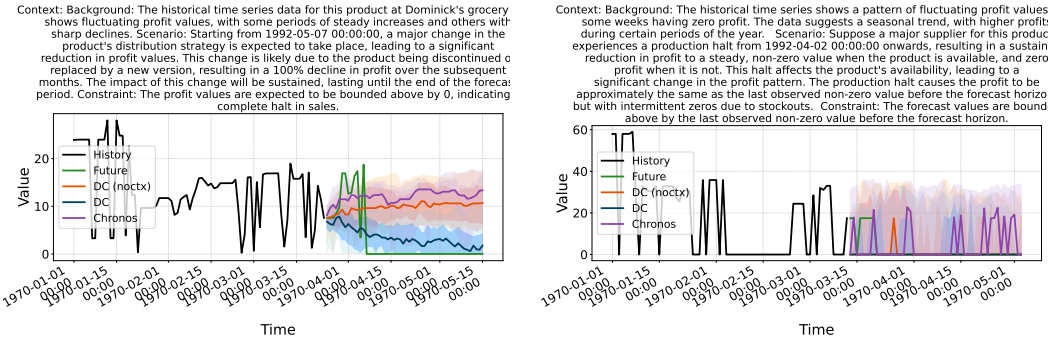


Figure 46: Examples of windows in zeroshot-test (EASY) where the context was found to be helpful when forecasting using DoubleCast.

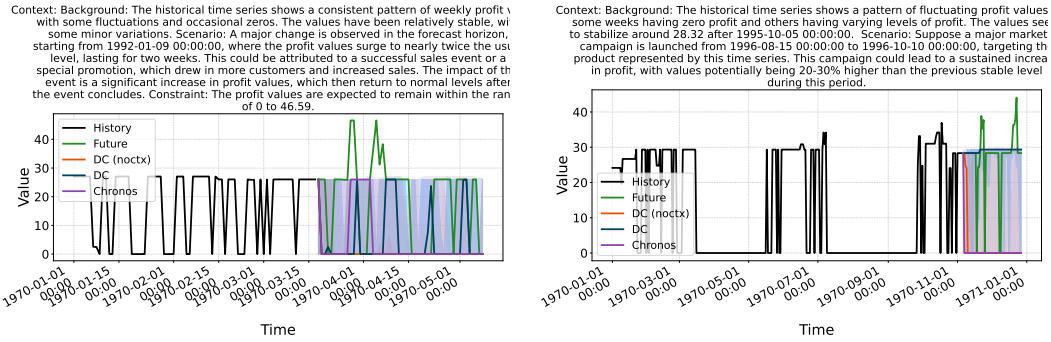


Figure 47: Examples of windows in zeroshot-test (EASY) where the context was found to be helpful when forecasting using DoubleCast.

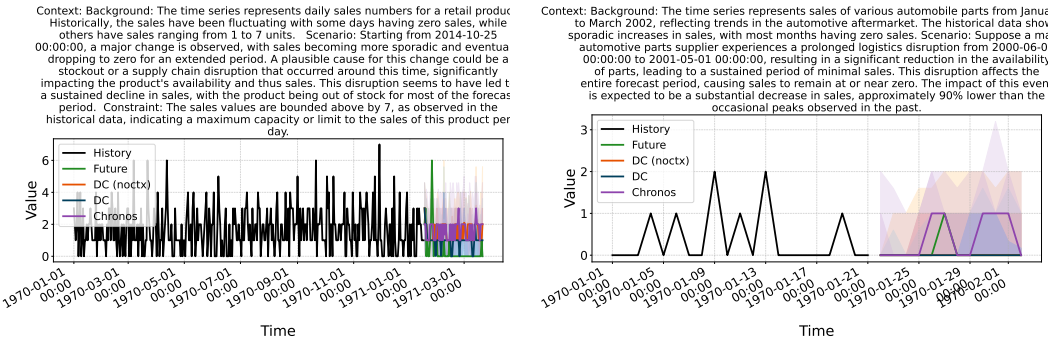


Figure 48: Examples of windows in zeroshot-test (EASY) where the context was found to be helpful when forecasting using DoubleCast.

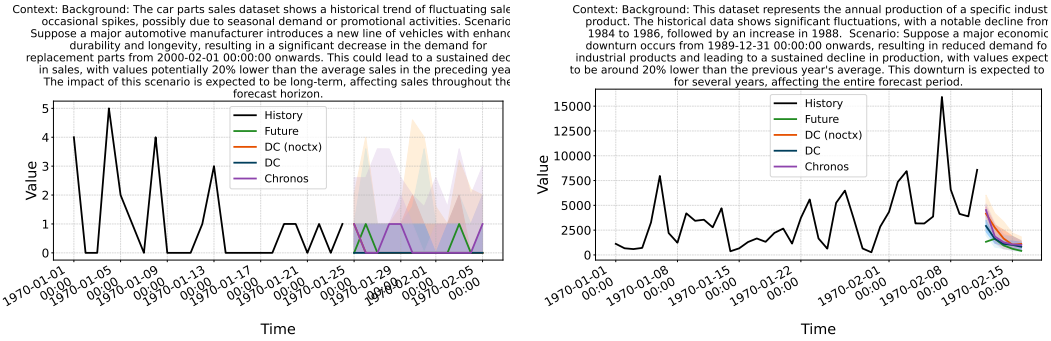


Figure 49: Examples of windows in `zeroshot-test` (EASY) where the context was found to be helpful when forecasting using DoubleCast.

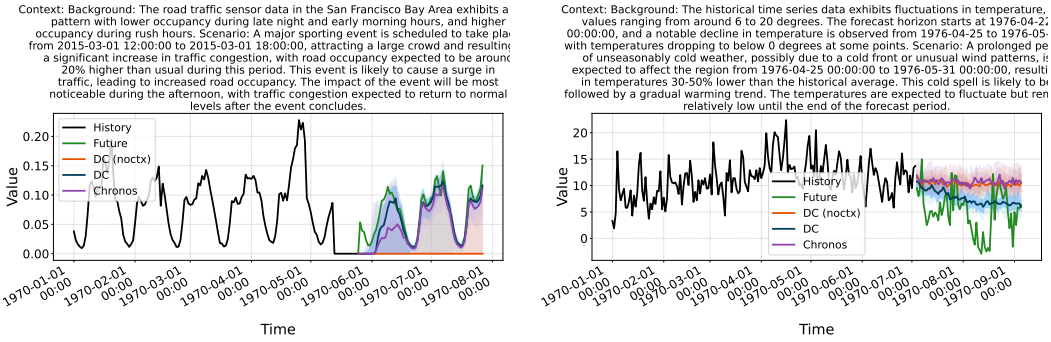


Figure 50: Examples of windows in `zeroshot-test` (EASY) where the context was found to be helpful when forecasting using DoubleCast.

G.6.4 `ZEROSHOT-TEST` (EASY): CONTEXT HURTS

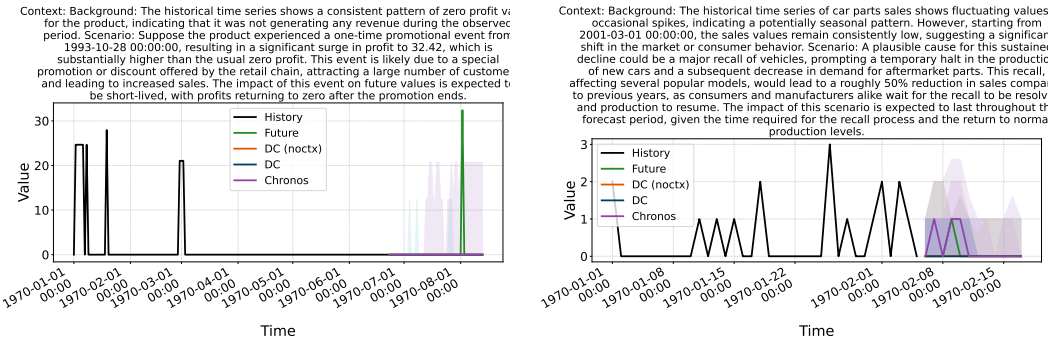


Figure 51: Examples of windows in `zeroshot-test` (EASY) where the context was found to not be helpful when forecasting using DoubleCast.

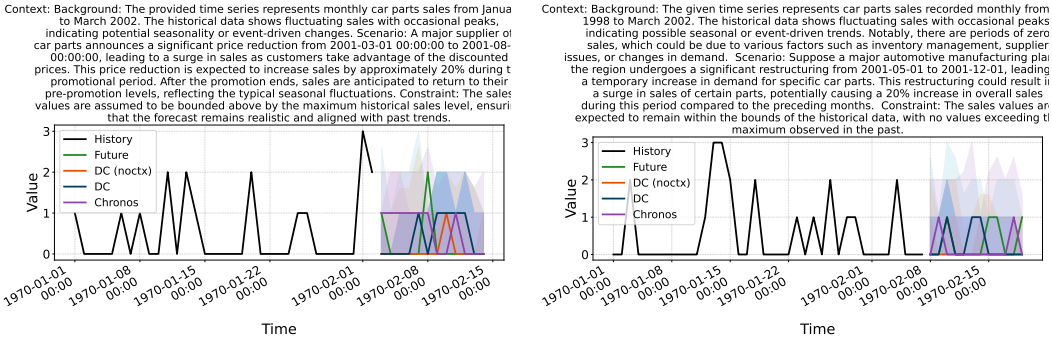


Figure 52: Examples of windows in `zeroshot-test` (EASY) where the context was found to not be helpful when forecasting using DoubleCast.

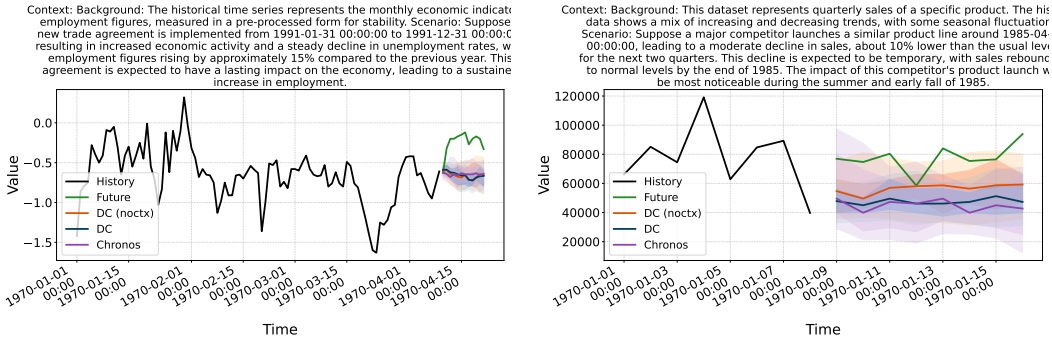


Figure 53: Examples of windows in `zeroshot-test` (EASY) where the context was found to not be helpful when forecasting using DoubleCast.

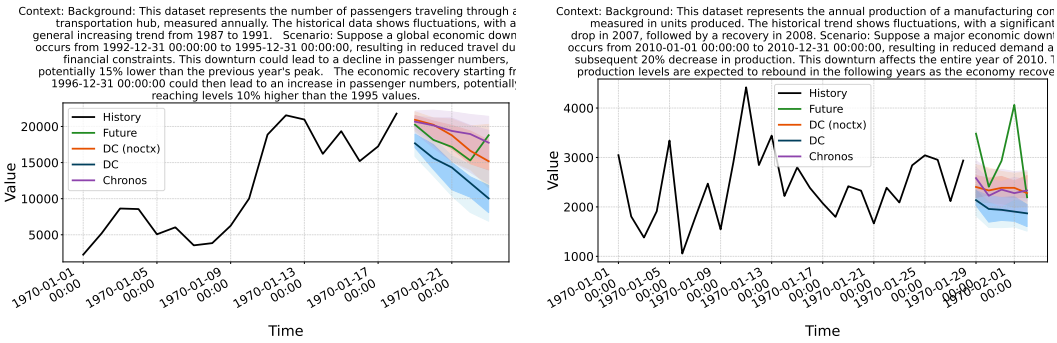


Figure 54: Examples of windows in `zeroshot-test` (EASY) where the context was found to not be helpful when forecasting using DoubleCast.

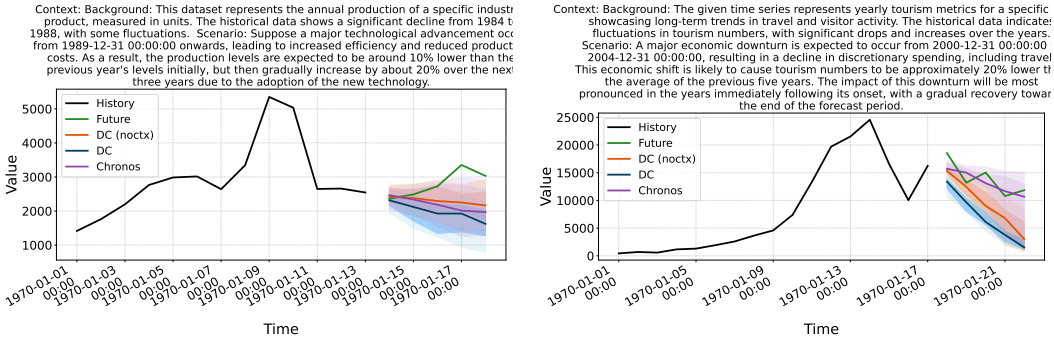


Figure 55: Examples of windows in `zeroshot-test` (EASY) where the context was found to not be helpful when forecasting using DoubleCast.