ROUTECAST: INSTANCE-DEPENDENT CONTEXT ROUTING FOR TIME SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Time-series foundation models (TSFMs) primarily operate on numerical sequences, yet the same observed history can be represented through alternative views that expose different pretrained inductive biases. However, the usefulness of these derived views need not be uniform across forecasting instances, making static multi-view integration potentially suboptimal. We introduce **RouteCast**, a parameter-efficient framework that formulates multi-view time-series forecasting as an *instance-dependent view relevance* problem. RouteCast separates *view inference* from *view utilization*: *Context Distillation* constructs instance-conditioned representations from visual and textual views derived solely from the observed history, while *Context Routing* dynamically regulates their relative influence on a frozen TSFM. All pretrained backbones remain frozen, with training restricted to lightweight adaptation modules. Across 12 forecasting benchmarks, RouteCast consistently improves over its underlying TSFM backbone and remains competitive with strong recent and controlled multi-view baselines. With Chronos as the backbone, RouteCast outperforms full fine-tuning on 11 of 12 datasets, while analogous backbone studies with Timer and TimesFM show similar improvements. Ablations further show complementary benefits from instance-conditioned distillation and adaptive routing. These results demonstrate the value of regulating alternative views according to their instance-dependent relevance for parameter-efficient TSFM adaptation.

1 INTRODUCTION

Recent advances in time series forecasting have been driven by *Time Series Foundation Models* (TSFMs), which leverage large-scale pretraining to learn transferable temporal representations and achieve strong zero-shot and few-shot generalization across diverse forecasting tasks (Ansari et al., 2024; Liu et al., 2024b). By learning generic temporal dynamics from large collections of time series, TSFMs substantially reduce the need for task-specific modeling and extensive feature engineering. Most TSFMs, however, operate primarily on numerical sequences. While numerical observations provide a precise representation of temporal values and dependencies, the same historical signal can also be expressed through alternative views that expose different structural characteristics. For example, a visual representation may expose shape- and pattern-level structure, whereas a textual representation can provide an explicit abstraction of higher-level properties such as trend and seasonality.

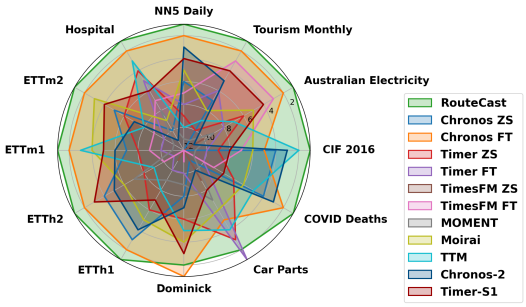


Figure 1: Radar plot showing per-dataset model rankings (1 = best) comparing RouteCast (Chronos) with strong time-series foundation model baselines under zero-shot (ZS) and full fine-tuned (FT) settings.

This perspective connects time series forecasting to *multi-view learning*, where multiple representations of the same underlying entity are used to expose complementary inductive biases (Xu et al., 2013; Zhao et al., 2017). Motivated by this perspective, we focus on a derived multi-view setting

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

in which all views are constructed solely from the same observed history and therefore do not incorporate future observations or independent exogenous information. Instead, they expose the same temporal evidence to pretrained models with different representational inductive biases. This distinction raises a more fundamental question than whether multiple views can simply be combined: *when should an alternative view of the same time-series history influence a forecast?*

The answer is unlikely to be uniform across forecasting instances. A visual abstraction may be useful when shape or periodic structure is particularly informative, whereas a textual summary may help when high-level trend information is more salient. Conversely, a derived view may provide little useful signal for another instance or may emphasize an abstraction that is not predictive of the future. Thus, complementarity at the representation level does not imply uniform predictive utility at the instance level. The central challenge is therefore not merely *how* to represent or combine multiple views, but *how much each view should influence each individual prediction*.

Recent work has begun to explore alternative representations of time series using pretrained language and vision models. Language-based approaches represent temporal patterns through prompts or language-model-based transformations (Jin et al., 2024; Liu et al., 2024a), while vision-based approaches transform time series into two-dimensional representations that can be processed by pretrained vision models (Chen et al., 2025). These studies demonstrate that pretrained models outside the time-series domain can provide useful representational biases for forecasting. However, existing approaches typically determine how auxiliary representations are constructed or fused at the model level, using fixed prompting, alignment, or predetermined integration strategies. They do not explicitly regulate the predictive influence of each derived view according to the current forecasting instance. As a result, a model may continue to rely on a view even when that view provides limited predictive utility for a particular input.

We therefore formulate multi-view time series forecasting as an *instance-dependent view relevance problem*. Rather than assuming that derived views are uniformly useful, a forecasting model should estimate their predictive relevance conditioned on the current forecasting instance. This formulation separates two questions that are often conflated in existing approaches: *what forecasting-relevant representation can be extracted from each view?* and *when should that representation be used?*

To address these questions, we propose **RouteCast**, a parameter-efficient framework for *instance-dependent multi-view routing* over a frozen TSFM. RouteCast explicitly separates view inference from view utilization through two complementary components. First, *Context Distillation* constructs instance-conditioned representations from visual and textual views derived from the same time-series history. Rather than requiring explicit alignment between heterogeneous view representations, it conditions each auxiliary representation on the current temporal instance to extract forecasting-relevant context. Second, *Context Routing* regulates how strongly these distilled views influence the TSFM as a function of the current temporal representation. This lets RouteCast to emphasize useful auxiliary views while reducing less relevant ones on an instance-by-instance basis.

Crucially, RouteCast introduces this adaptive multi-view capability without modifying the pretrained foundation models themselves. The vision encoder, text encoder, and TSFM backbone remain frozen, while learning is restricted to lightweight modules for context distillation, projection, and routing. This design preserves the pretrained temporal capabilities of the TSFM while allowing it to exploit alternative representations through input-dependent control. Across diverse forecasting benchmarks, RouteCast improves over strong zero-shot and fully fine-tuned TSFM baselines as well as static multi-view alternatives. Component ablations further show instance-conditioned distillation and adaptive routing provide complementary benefits beyond simply introducing additional views.

In summary, our contributions are threefold. First, we formulate **multi-view time series forecasting as an instance-dependent view relevance problem**, recognizing that alternative representations of the same observed history can exhibit different predictive utility across forecasting instances. Second, we introduce **RouteCast**, a parameter-efficient framework that separates *view inference* from *view utilization* through Context Distillation and Context Routing over a frozen TSFM, enabling adaptive control of derived visual and textual views without requiring an explicit cross-view alignment objective. Finally, we provide comprehensive experiments across diverse forecasting benchmarks showing that instance-dependent multi-view routing improves over zero-shot and fully fine-tuned TSFMs, static multi-view integration, and alternative model configurations, while requiring updates only to lightweight adaptation modules.

2 Related Work

Foundation models for time-series forecasting. Recent time-series foundation models (TSFMs), including Timer (Liu et al., 2024b), TimesFM (Das et al., 2024), TTM (Ekambaram et al., 2024), Chronos (Ansari et al., 2024), MOMENT (Goswami et al., 2024), and MOIRAI (Woo et al., 2024), are pretrained on large-scale time-series corpora and demonstrate transfer across forecasting domains. Their success suggests that pretrained temporal representations provide a strong numerical foundation. RouteCast builds on this foundation and asks whether alternative representations of the same observed history can provide complementary context without updating the pretrained TSFM.

Language- and vision-based time-series forecasting. Language-based approaches adapt pretrained language models for forecasting by converting time series into textual representations or conditioning prediction on prompts, including Time-LLM (Jin et al., 2024), LangTime (Niu et al., 2025), ChatTime (Wang et al., 2025), and TaTS (Li et al., 2026). These methods show that language-model priors can be useful for temporal prediction, but typically use a fixed textual representation or prompt structure for all instances. Vision-based approaches render time series as images and exploit pretrained vision encoders (Chen et al., 2025; Yang et al., 2025; Shen et al., 2025). They provide an alternative route for transferring visual pattern-recognition priors to time series, but generally treat the visual representation as the primary forecasting input. RouteCast retains the numerical series as the forecasting backbone and uses visual and textual views as complementary, derived context.

Cross-modal and multimodal forecasting. Several approaches jointly model time-series and language or vision through shared-space alignment or coupled multimodal representations. TimeCMA and VoT align time-series and text embeddings (Liu et al., 2025b; Wang et al., 2026), while Time-VLM incorporates vision–language models into time-series reasoning (Zhong et al., 2025). These methods demonstrate the value of cross-representation modeling, but do not directly study how the usefulness of each derived view may vary across forecasting instances. A related line addresses *exogenous multimodal forecasting*, where external signals such as news, reports, or metadata are combined with numerical observations. Aurora (Wu et al., 2026), CAPTime (Yao et al., 2025), and AIR (Seo et al., 2024) use such external data streams to improve prediction. In contrast, RouteCast considers a self-contained *derived multi-view* setting: all visual and textual views are constructed directly from the target numerical history. This removes the need for external domain-specific data and isolates the question of how to use multiple representations of the same signal.

Overall, existing approaches rely on fixed, or implicitly coupled context integration strategies. None of these methods support explicit instance-dependent routing over contextual signals, limiting their ability to adaptively emphasize or suppress multiple views based on input-specific relevance.

Positioning and Contributions Table 5 summarizes existing approaches and positions our work along four key dimensions: conditioning, context usage, instance control, and model freezing. In terms of conditioning, prior methods either directly transfer inductive biases from other modalities, align representations within a shared space, or rely on prompts to implicitly induce adaptation. Regarding context usage, existing approaches typically employ fixed representations that remain invariant to the input context, or coupled representations, where the influence of each view is entangled. Furthermore, they lack explicit instance-level control over how much each view contributes to an individual input, making such contributions difficult to regulate dynamically. Finally, methods that do not keep the underlying model frozen cannot fully leverage the generalization capabilities of TSFMs, which are pretrained on large-scale numerical time-series data. In contrast, RouteCast addresses these limitations through *instance-dependent context routing*, which adaptively determines the influence of each view conditioned on the input. By separating context inference from utilization while keeping the pretrained backbones frozen, RouteCast enables parameter-efficient and adaptive control over multi-view contextual information.

3 Method

Design Philosophy We formulate multi-view time-series forecasting as a problem of *instance-dependent view relevance*. The visual and textual views considered in this work are derived from the same observed time-series history and therefore do not provide additional future information. Instead, they expose the same temporal evidence through pretrained models with different represen-

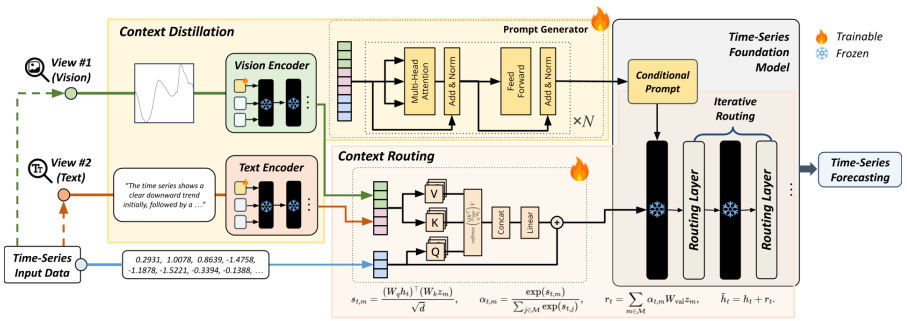


Figure 2: Overview of RouteCast for instance-dependent multi-view time-series forecasting. (Left) A single observed time-series history is represented through numerical, visual, and textual views, which are processed using frozen pretrained backbones. (Middle) Context Distillation constructs instance-conditioned representations from the derived visual and textual views through conditional soft prompting. (Right) Context Routing repeatedly estimates the relative relevance of the distilled auxiliary views conditioned on the evolving temporal representation and adaptively regulates their influence on the frozen TSFM. Only lightweight distillation, projection, and routing modules are trained. A conceptual comparison of the individual components is provided in Appendix F.

tational inductive biases. The predictive usefulness of these views, however, need not be uniform across forecasting instances.

Existing approaches (Cao et al., 2024; Jin et al., 2024; Liu et al., 2025a) typically rely on fixed prompts, static representations, or predetermined integration mechanisms, and therefore do not explicitly control how strongly each derived view influences an individual forecast. RouteCast addresses this limitation by separating two complementary operations: *view inference*, which extracts forecasting-relevant representations from each derived view conditioned on the current time-series instance, and *view utilization*, which regulates their influence on the frozen TSFM. These operations are implemented through *Context Distillation* and *Context Routing*, respectively.

Problem Setup and Inputs Let $X \in \mathbb{R}^{C \times D}$ denote an observed time-series segment with context length C and D variables, and let $Y \in \mathbb{R}^{H \times D}$ denote the future values over forecasting horizon H. Our objective is to learn a forecasting function $\hat{Y} = f(X, V, L)$, where $V = g_v(X)$ and $L = g_t(X)$ denote visual and textual views derived from the same observed history X.

The three inputs therefore represent different views of the same forecasting instance rather than independent sources of future information. The numerical view is converted into patch-level representations using the preprocessing of the underlying TSFM, while V and L are processed by frozen pretrained vision and text encoders, respectively. RouteCast operates only through lightweight adaptation modules around these frozen pretrained models; the vision encoder, text encoder, and TSFM backbone are never updated during training.

For notational simplicity, we describe the setting with one visual and one textual auxiliary view, although the routing formulation naturally extends to additional views. The complete computation flow is summarized in Algorithm 1 in Appendix E.

RouteCast Framework RouteCast augments a frozen TSFM with instance-dependent multi-view adaptation while keeping all pretrained backbones fixed. Given numerical, visual, and textual views of the same observed history, RouteCast first constructs instance-conditioned auxiliary representations using *Context Distillation*. It then applies *Context Routing* to regulate the influence of these representations on the temporal hidden states of the TSFM. Only the prompt generators, projection layers, and routing module are trainable. This decoupling allows RouteCast to exploit representations produced by heterogeneous pretrained models without requiring an explicit cross-view alignment objective or end-to-end backbone fine-tuning.

Context Distillation Context Distillation constructs forecasting-relevant auxiliary representations that are conditioned on the current time-series instance. Rather than directly feeding fixed visual

or textual representations into the TSFM, we use conditional soft prompts to adapt how the frozen auxiliary encoders process each derived view. The procedure consists of two steps.

(1) Instance-conditioned view refinement. Let O_v, O_t, and O_x denote the input-level visual patch representations, text token representations, and time-series patch representations, respectively. We first summarize these representations and project them into the input space of each auxiliary prompt generator. For auxiliary view $m \in \{v, t\}$, a Transformer-based prompt generator produces an instance-specific soft prompt

$$c_m = G_m\left(\text{Concat}\left[P_{v\to m}(\bar{O}_v), P_{t\to m}(\bar{O}_t), P_{x\to m}(\bar{O}_x)\right]\right), \tag{1}$$

where $\bar{O}_i$ denotes a pooled summary of input view i, $P_{i\to m}$ denotes a projection into the representation space required by prompt generator G_m when necessary, and G_m denotes the corresponding prompt generator. The generated prompt is prepended to the corresponding auxiliary input and processed by the frozen encoder:

$$E_v = \text{Pool}_v\left(F_v([c_v; O_v])\right), \qquad E_t = \text{Pool}_t\left(F_t([c_t; O_t])\right), \tag{2}$$

where F_v and F_t are the frozen vision and text encoders. We use the first token of the final hidden state for the visual representation and average pooling for the textual representation; alternative pooling strategies are studied in Appendix M.

Because c_v and c_t depend on the current time-series instance, the resulting representations are not fixed embeddings of the derived views but instance-conditioned representations tailored to the current forecasting instance.

(2) TS-conditioned prompt generation. The refined auxiliary representations are projected into the TSFM feature space and combined with a summary of the numerical view:

$$p_x = G_x\left(\text{Concat}[P_{v\to x}(E_v), P_{t\to x}(E_t), \bar{O}_x]\right). \tag{3}$$

The resulting soft prompt p_x conditions the frozen TSFM on the distilled multi-view context while preserving the view-specific auxiliary representations for subsequent routing.

Context Routing Context Distillation determines *what* forecasting-relevant representation is extracted from each auxiliary view, whereas Context Routing determines *how strongly* each distilled view influences the current temporal representation.

Let $\mathcal{M} = \{\text{vis}, \text{text}\}$ denote the set of auxiliary views, and let $z_m \in \mathbb{R}^d$ denote the distilled representation of view $m \in \mathcal{M}$ after projection into the TSFM feature space. For a temporal hidden state h_t at position t, we compute a view-specific compatibility score

$$s_{t,m} = \frac{(W_q h_t)^\top (W_k z_m)}{\sqrt{d}}, \qquad m \in \mathcal{M}. \tag{4}$$

The scores are normalized across auxiliary views:

$$\alpha_{t,m} = \frac{\exp(s_{t,m})}{\sum_{j\in\mathcal{M}} \exp(s_{t,j})}, \tag{5}$$

where $\alpha_{t,m}$ represents the relative routing weight assigned to view m at temporal position t.

The routed auxiliary representation is then

$$r_t = \sum_{m\in\mathcal{M}} \alpha_{t,m} W_{\text{val}} z_m, \tag{6}$$

and is incorporated into the temporal state through a residual update,

$$\tilde{h}_t = h_t + r_t. \tag{7}$$

Routing is applied repeatedly as the temporal representations evolve through the frozen TSFM. At layer ℓ, the routing weights are conditioned on the current hidden representation $h_t^{(\ell)}$, allowing the relative contribution of the visual and textual views to vary across forecasting instances, temporal positions, and representation depth. The same routing parameters are shared across TSFM layers to limit trainable capacity. Importantly, the routing weights are not treated as fixed view coefficients. They are recomputed from the current temporal representation and therefore provide an input-dependent allocation of auxiliary influence. Unlike static fusion, this mechanism allows RouteCast to increase the relative contribution of a view when its distilled representation is compatible with the current temporal state and reduce its contribution otherwise.

4 EVALUATION SETUP

Datasets and Metrics[1]

Following the evaluation protocol of Chronos (Ansari et al., 2024), we evaluate RouteCast on a representative subset of Benchmark II spanning diverse domains, sampling frequencies, and forecasting regimes. The resulting benchmark covers heterogeneous temporal characteristics under which the predictive utility of different derived views may vary across forecasting instances. This enables us to evaluate whether instance-dependent view utilization provides benefits over fixed integration strategies. For the primary Chronos-based evaluation, we select datasets reported as excluded from the Chronos pretraining corpus in order to minimize known overlap between pretraining and downstream evaluation. All compared methods use identical forecasting configurations and data splits within each dataset. Because the pretraining corpora of different TSFMs are not necessarily identical, we do not assume that the same exclusion guarantee holds universally for every baseline backbone. We use Mean Squared Error (MSE) as the primary forecasting metric. Since absolute MSE scales differ substantially across datasets, we report performance separately for each dataset and use aggregate ranking and relative performance summaries for cross-dataset comparison rather than relying solely on the raw mean of MSE values. Unless otherwise specified, RouteCast keeps the pretrained vision encoder, text encoder, and TSFM backbone frozen, with training restricted to its lightweight adaptation components.

Table 1: Summary of the 12 evaluation datasets. Detailed descriptions and preprocessing procedures are provided in Appendix C.

Dataset	Freq.	C	H	# Series
NN5 Daily	1 D	56	56	111
Aus. Elec.	30 min	48	48	5
Tourism Monthly	1 M	24	24	366
COVID Deaths	1 D	30	30	266
Car Parts	1 M	12	12	2674
CIF 2016	1 M	12	12	72
Dominick	1 W	8	8	100014
Hospital	1 M	12	12	767
ETTh1	1 hr	24	24	14
ETTh2	1 hr	24	24	14
ETTm1	15 min	96	24	14
ETTm2	15 min	96	24	14

Baselines. We compare RouteCast against two complementary groups of baselines: strong pretrained time-series models and controlled multi-view forecasters.

Time-series forecasting baselines. We include widely used TSFMs spanning different architectural and pretraining paradigms: Chronos (Ansari et al., 2024), Chronos-2 (Ansari et al., 2025), Timer (Liu et al., 2024b), Timer-S1 (Liu et al., 2026), TimesFM (Das et al., 2024), MOMENT (Goswami et al., 2024), MOIRAI (Woo et al., 2024), and TTM (Ekambaram et al., 2024). Chronos-2 and Timer-S1 are included as a recent successor to Chronos and Timer in order to evaluate RouteCast against a stronger and more recent generation of TSFM baselines. For Chronos, Timer, and TimesFM, which are also used as alternative TSFM backbones within RouteCast, we additionally compare against both the standalone zero-shot (ZS) model and its fully fine-tuned (FT) counterpart. These same-backbone comparisons are particularly important because they isolate whether RouteCast provides benefits beyond the pretrained temporal backbone itself and beyond conventional end-to-end fine-tuning. Chronos-2 and Timer-S1 are used as an additional strong external baseline rather than as a RouteCast backbone in the main experiments.

Controlled multi-view baselines. To determine whether RouteCast's gains can be reproduced by simply adding derived visual and textual views, we construct a controlled 2×2 family of multi-view forecasters. The two factors are: (i) auxiliary encoder initialization, comparing encoders trained from scratch against pretrained vision/text encoders matching RouteCast; and (ii) forecasting architecture, comparing MLP-based and Transformer-based integration. This produces four variants: $\text{MLP}_{\text{scratch}}$, $\text{MLP}_{\text{pretrained}}$, $\text{Transformer}_{\text{scratch}}$, and $\text{Transformer}_{\text{pretrained}}$. All four baselines receive the same numerical, visual, and textual views and use the same input preprocessing as RouteCast, but do not employ a pretrained TSFM backbone. These comparisons help separate three possible sources of performance improvement: the pretrained temporal backbone, pretrained auxiliary representations, and the mechanism used to integrate multiple derived views. Implementation details for all baseline models are provided in Appendix H.

Implementation Details are documented in Appendix G.

[1] Additional details, including dataset statistics and preprocessing procedures, are provided in Appendix C.

5 RESULT

5.1 OVERALL PERFORMANCE

Table 2: Forecasting MSE on twelve datasets. RouteCast constructs additional views from the observed history using CLIP and Qwen, while keeping all pretrained time-series, vision, and text backbones frozen. We compare against zero-shot and fine-tuned TSFMs as well as controlled multi-view baselines. **Bold** and underlined values denote the best and second-best results within each dataset. Full five-run results with standard deviations are provided in Appendix I.

	NN5	Aus	Tour	COV	Car	CIF	Dom	Hos	ETT				Avg
									h1	h2	m1	m2	
Chronos♠	0.791	2.670	2.468	0.156	1.303	7.461	1.340	2.613	0.289	0.172	0.128	0.083	1.623
Chronos♡	0.481	0.385	1.367	0.127	0.997	7.488	**0.952**	2.364	0.273	0.144	0.106	0.037	1.227
Timer♠	1.099	1.462	2.622	0.218	0.957	7.597	1.194	2.455	0.315	0.282	0.139	0.084	1.535
Timer♡	1.097	1.539	2.466	0.385	0.919	7.727	2.148	2.470	0.672	0.647	0.241	0.220	1.711
TimesFM♠	1.759	2.282	3.170	1.208	1.073	8.075	3.733	2.634	1.895	1.896	1.998	2.015	2.645
TimesFM♡	0.847	0.476	1.634	0.343	3.370	7.520	2.408	2.526	1.949	0.436	0.239	0.193	1.828
MOMENT♠	1.130	1.576	2.690	0.517	1.052	7.748	2.026	2.562	0.368	0.275	0.245	0.222	1.701
MOIRAI♠	0.529	1.199	2.596	0.189	1.046	7.498	1.057	2.574	0.307	0.243	0.122	0.063	1.452
TTM♠	1.117	1.512	2.602	0.176	0.991	7.384	1.074	2.448	0.355	0.358	0.118	0.121	1.521
Chronos-2♠	0.519	2.291	2.241	0.132	1.132	7.411	1.206	2.621	0.290	0.181	0.126	0.088	1.520
Timer-S1♠	0.525	1.149	2.239	0.250	1.061	7.568	1.005	2.474	0.337	0.169	0.124	0.077	1.415
MLP$_{\text{scratch}}$	0.481	0.430	1.355	0.072	**0.908**	7.394	1.219	**2.317**	0.278	0.148	0.102	0.042	1.229
Transformer$_{\text{scratch}}$	0.482	0.429	1.355	0.072	**0.908**	7.392	1.219	2.318	0.278	0.149	0.102	0.041	1.229
MLP$_{\text{pretrained}}$	0.481	0.432	**1.354**	**0.071**	**0.908**	7.392	1.220	2.318	0.278	0.149	0.102	0.041	1.229
Transformer$_{\text{pretrained}}$	0.480	0.431	1.355	0.072	**0.908**	7.394	1.210	**2.317**	0.278	0.149	0.102	0.041	1.228
RouteCast(Ours)	**0.465**	**0.332**	1.359	0.107	0.951	**7.329**	0.971	2.322	**0.266**	**0.136**	**0.095**	**0.032**	**1.197**

♠ Zero-shot. ♡ Fine-tuned.

Table 2 compares RouteCast with time-series foundation models (TSFMs), their fully fine-tuned counterparts, and controlled multi-view baselines across twelve forecasting datasets. RouteCast achieves the lowest MSE on seven datasets: NN5, Australian Electricity, CIF, and all four ETT variants. It is also second-best on the Dominick dataset. In particular, RouteCast improves over fine-tuned Chronos on 11 of 12 datasets, despite keeping the pretrained time-series, vision, and text backbones frozen and training Context Distillation and Context Routing modules only.

The comparison with fine-tuned TSFMs shows that these improvements cannot be attributed solely to updating the forecasting backbone on target data. Relative to fine-tuned Chronos, RouteCast reduces MSE from 0.481 to 0.465 on NN5 and from 0.385 to 0.332 on Australian Electricity. It also improves consistently across the ETT suite, reducing MSE from 0.273, 0.144, 0.106, and 0.037 to 0.266, 0.136, 0.095, and 0.032 on ETTh1, ETTh2, ETTm1, and ETTm2, respectively. These results show that conditioning a frozen forecaster on complementary views derived from the observed history can be competitive with, and often preferable to, end-to-end adaptation.

The controlled multi-view baselines distinguish the benefit of exposing additional views from that of adaptively integrating them. The MLP- and Transformer-based baselines are competitive on several datasets, and their performance differs little between pretrained and scratch initialization. However, RouteCast obtains the lowest reported average MSE and outperforms all controlled multi-view alternatives throughout the ETT suite. This pattern suggests that view availability alone does not account for the full gain: constructing instance-specific context from each view and selectively routing that context provides additional benefit beyond fixed fusion.

RouteCast is not uniformly best on every dataset. Controlled multi-view baselines remain strongest on Tourism, COVID-19, Car Parts, and Hospital, while fine-tuned Chronos performs best on the Dominick dataset. Thus, the advantage of RouteCast is broad but dataset-dependent. Full results over five runs, including standard deviations, are reported in Appendix I.

5.2 EFFECT OF PRETRAINED BACKBONES AND ENCODERS

We first test whether RouteCast is tied to a particular forecasting backbone. Table 3 instantiates the same framework on three pretrained TSFMs (Chronos, Timer, and TimesFM) and compares each

frozen backbone with its zero-shot (ZS) and fully fine-tuned (FT) variants. Across the three datasets shown, RouteCast improves over both ZS and FT for every backbone. For Chronos, it reduces the average MSE from 0.744 under full fine-tuning to 0.719 while keeping the backbone frozen. The corresponding average MSE is reduced from 1.701 to 0.711 for Timer and from 0.986 to 0.704 for TimesFM. The result is particularly informative for Timer and TimesFM, whose fine-tuned variants remain substantially weaker than RouteCast despite updating all backbone parameters. Thus, the benefit is not specific to the pretraining objective or architecture of Chronos, nor does it require end-to-end adaptation of the TSFM. Across these tested backbones, RouteCast provides a consistent way to condition a frozen forecaster on derived views through Context Distillation and Context Routing.

Table 3: Effect of Pretrained TSFM Backbones.

TS	VE	TE	NN5	Aus	Tour	Avg
Chronos	ZS		0.791	2.670	2.468	1.976
	FT		0.481	0.385	1.367	0.744
	CLIP	Qwen	**0.465**	**0.332**	**1.359**	**0.719**
Timer	ZS		1.099	1.462	2.622	1.728
	FT		1.097	1.539	2.466	1.701
	CLIP	Qwen	**0.496**	**0.291**	**1.345**	**0.711**
TimesFM	ZS		1.759	2.282	3.170	2.404
	FT		0.847	0.476	1.634	0.986
	CLIP	Qwen	**0.470**	**0.300**	**1.343**	**0.704**

Table 4: Effect of vision and text encoders.

	VE	TE	NN5	Aus	Tour	Avg
	ZS		0.791	2.670	2.468	1.976
	FT		0.481	0.385	1.367	0.744
RouteCast (Ours)	CLIP	×	0.470	0.335	1.364	0.723
	BLIP	×	0.469	0.369	1.362	0.733
	×	Qwen	0.706	0.421	1.406	0.844
	×	Llama	0.541	0.401	1.365	0.769
	CLIP	Qwen	0.465	0.332	1.359	0.719
	BLIP	Qwen	0.464	0.325	1.361	0.717
	CLIP	Llama	0.463	0.326	1.354	0.714
	BLIP	Llama	**0.459**	**0.322**	**1.352**	**0.711**

We next examine the role of the view encoders and whether the gains require a particular vision–language model pair. Table 4 uses Chronos as the forecasting backbone and varies the vision encoder (CLIP or BLIP) and text encoder (Qwen or Llama). Vision-only RouteCast already improves over fine-tuned Chronos on all three datasets for both CLIP and BLIP. Text-only configurations also improve substantially over the zero-shot baseline, although they are weaker than vision-only configurations in this setting. Combining the two views yields a further improvement: every vision–text configuration achieves a lower average MSE than either of its corresponding single-view configurations, and all combined configurations improve on fine-tuned Chronos for NN5, Australian Electricity, and Tourism. The best encoder pair in this comparison is BLIP with Llama, which obtains an average MSE of 0.711, compared with 0.744 for fine-tuned Chronos. However, the remaining combined configurations are close (0.714–0.719), and all outperform the time-series-only ZS and FT baselines. This consistency suggests that RouteCast does not rely on one specific vision or text encoder; rather, the two derived views provide complementary context that the routing mechanism can exploit. Full three-run results, including standard deviations, are reported in Appendix J.

5.3 Impact of Context Distillation and Routing

We ablate Context Distillation (CD) and Context Routing (CR) to distinguish the contribution of instance-conditioned context construction from that of adaptive view integration. The *None* variant replaces CD with static soft prompts and CR with heuristic fusion. The CD-only variant retains instance-conditioned context construction but uses heuristic fusion, whereas the CR-only variant applies routing to static prompts. The full model combines both components. Figure 3 shows that the full RouteCast configuration achieves the lowest MSE on all three datasets. Relative to the *None* variant, adding CR alone improves performance on NN5, Australian Electricity, and Tourism, while adding CD alone improves Australian Electricity and Tourism but not NN5. Thus, neither component alone is uniformly optimal across datasets. However, combining CD

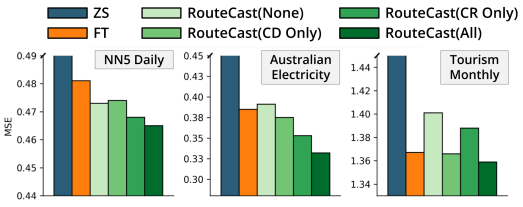


Figure 3: Ablation of Context Distillation (CD) and Context Routing (CR). *None* uses static soft prompts and heuristic fusion; CD-only uses instance-conditioned context with heuristic fusion; CR-only applies routing to static prompts; and *All* combines CD and CR. The full configuration achieves the lowest MSE on all three datasets. The single components provide dataset-dependent gains, whereas their combination improves on both single-component variants throughout the comparison.

and CR improves on both single-component variants in every case. This pattern supports the intended division of labor between the two components. CD constructs view-specific context conditioned on the input history, whereas CR adaptively integrates information from the derived views into the forecasting representation. The full model benefits when routing operates on instance-conditioned context rather than static prompts, and when the distilled context is integrated through routing rather than heuristic fusion. Exact numerical results are provided in Appendix K; additional implementation variants and routing analyses appear in Appendices L and Q, respectively.

6 Qualitative Analysis

We complement the aggregate results with representative forecasting examples that compare RouteCast with fine-tuned Chronos. These examples are intended to illustrate the types of trajectory differences associated with the quantitative results, rather than to provide a causal attribution of individual predictions to a particular view.

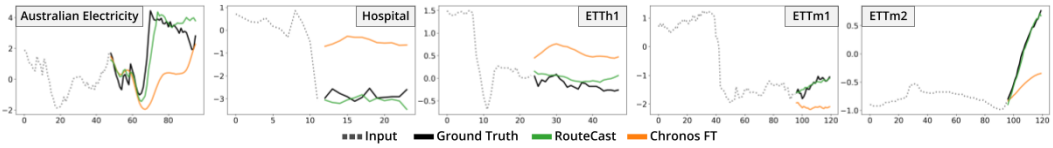


Figure 4: **Representative forecasting examples.** Each panel compares the prediction of fine-tuned Chronos (*Chronos FT*) and RouteCast (*RouteCast*) with the ground truth. The dotted curve denotes the observed input context, and colored curves show predictions over the forecast horizon.

Forecasting examples. Figure 4 shows representative examples from Australian Electricity, Hospital, ETTh1, ETTm1, and ETTm2. In these cases, RouteCast more closely tracks the target trajectory than fine-tuned Chronos, including changes in level, local turning behavior, and forecast amplitude. For example, RouteCast better follows the post-context evolution in the Australian Electricity and ETT examples, where the fine-tuned baseline either smooths the trajectory or remains biased toward its extrapolated trend.

These examples should be interpreted together with the aggregate evaluation: RouteCast outperforms fine-tuned Chronos on Australian Electricity and all four ETT datasets in Table 2. They provide an intuitive illustration of those dataset-level improvements, but do not imply that RouteCast is preferable for every individual forecasting window or that a particular derived view causes a specific prediction. Additional examples, including cases with smaller gains, are provided in Appendix R.

7 Conclusion

We introduced RouteCast, a parameter-efficient framework for multi-view time-series forecasting from alternative representations derived from the same observed history. RouteCast separates *Context Distillation*, which constructs instance-conditioned context from visual and textual views, from *Context Routing*, which adaptively integrates this context into a frozen time-series forecaster. Across twelve forecasting datasets, RouteCast achieves the lowest MSE on seven datasets and improves over fine-tuned Chronos on 11 of 12 datasets while updating only lightweight modules. Further analyses show that these gains extend across several TSFM backbones and encoder pairs, and that combining Context Distillation with Context Routing is consistently stronger than either component alone in the ablation setting. These results support derived multi-view forecasting as an instance-dependent view-utilization problem: supplying additional views alone is not sufficient, and the value of those views depends on how they are converted into context and integrated with the temporal representation. RouteCast therefore offers a practical way to extend pretrained TSFMs without end-to-end backbone fine-tuning. Future work can examine richer derived views and characterize when particular views are most useful for a given forecasting instance.

AI USE STATEMENT

In this work, we used generative AI tools for generating synthetic data sets (specifically, generating time-series descriptions using the Qwen3VL model, as detailed in Appendix D) and assisting with translation. We have not used generative AI tools for the other required disclosure tasks (developing theoretical models or conceptual frameworks, formulating or proving mathematical claims, writing proofs, proposing or refining hypotheses, designing research methodology or experiments, cleaning and reformatting datasets, supporting qualitative and thematic data analysis, or interpreting results). Additionally, we used generative AI tools for creating or editing software code and editing the research paper to improve readability. We have thoroughly reviewed all AI-assisted work, including verifying LLM-generated code for correctness and carefully validating the accuracy and quality of translated text, edited writing, and synthetic time-series descriptions. We take full responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our work, we have taken several key measures across our code, experimental protocols, and implementation details:

- **Codebase & Implementation:** We provide our complete source code, including implementation details for our proposed method. (anonymized repository: `https://anonymous.4open.science/r/RouteCast-E268`). Detailed setup instructions and baseline configurations are documented in Section 4 and Appendix H.
- **Experimental Rigor & Seeds:** To account for variance, all primary experiments were conducted across 5 independent random seeds, while secondary experiments were evaluated using 3 seeds or a single fixed seed. Detailed reporting on the number of runs per experiment, along with full numerical results containing means and standard deviations, is provided in Appendix I, J and K.

REFERENCES

{Abdul Fatir} Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, {Syama Sundar} Rangapuram, {Sebastian Pineda} Arango, Shubham Kapoor, Jasper Zschiegner, {Danielle C.} Maddix, Hao Wang, {Michael W.} Mahoney, Kari Torkkola, {Andrew Gordon} Wilson, Michael Bohlke-Schneider, and Yuyang Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024, 2024. ISSN 2835-8856. Publisher Copyright: © 2024, Transactions on Machine Learning Research. All rights reserved.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting, 2025. URL https://arxiv.org/abs/2510.15821.

Defu Cao, Furong Jia, Sercan Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. Tempo: Prompt-based generative pre-trained transformer for time series forecasting. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 18546–18578, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/5132940b1bced8a7b28e9695d49d435a-Paper-Conference.pdf.

Mouxiang Chen, Lefei Shen, Zhuo Li, Xiaoyun Joy Wang, Jianling Sun, and Chenghao Liu. VisionTS: Visual masked autoencoders are free-lunch zero-shot time series forecasters. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8979–9007. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chen25be.html.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H. Nguyen, Wesley M. Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 74147–74181. Curran Associates, Inc., 2024. doi: 10.52202/079017-2359. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/874a4d89f2d04b4bcf9a2c19545cf040-Paper-Conference.pdf.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. MOMENT: A family of open time-series foundation models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 16115–16152. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/goswami24a.html.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-llm: Time series forecasting by reprogramming large language models. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 23857–23880, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/680b2a8135b9c71278a09cafb605869e-Paper-Conference.pdf.

Junnan Li, Dongxu Li, Caiming Xiong, and Steven Hoi. BLIP: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Pro-*

ceedings of the 39th International Conference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pp. 12888–12900. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/li22n.html.

Zekun Li, Shiyang Li, and Xifeng Yan. Time series as images: Vision transformer for irregularly sampled time series. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 49187–49204. Curran Associates, Inc., 2023. doi: 10.52202/075280-2139. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/9a17c1eb808cf012065e9db47b7ca80d-Paper-Conference.pdf.

Zihao Li, Xiao Lin, Zhining Liu, Jiaru Zou, Ziwei Wu, Lecheng Zheng, Dongqi Fu, Yada Zhu, Hendrik Hamann, Hanghang Tong, and Jingrui He. Language in the flow of time: Time-series-paired texts weaved into a unified temporal narrative. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 24437–24484, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/29de5722d7d2a6946c08d0c0162a1c71-Paper-Conference.pdf.

Chanjuan Liu, Shengzhi Wang, and Enqiang Zhu. Dp-gpt4mts: Dual-prompt large language model for textual-numerical time series forecasting, 2025a. URL https://arxiv.org/abs/2508.04239.

Chenxi Liu, Qianxiong Xu, Hao Miao, Sun Yang, Lingzheng Zhang, Cheng Long, Ziyue Li, and Rui Zhao. Timecma: towards llm-empowered multivariate time series forecasting via cross-modality alignment. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025b. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i18.34067. URL https://doi.org/10.1609/aaai.v39i18.34067.

Haoxin Liu, Zhiyuan Zhao, Jindong Wang, Harshavardhan Kamarthi, and B. Aditya Prakash. LSTPrompt: Large language models as zero-shot time series forecasters by long-short-term prompting. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 7832–7840, Bangkok, Thailand, August 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.466. URL https://aclanthology.org/2024.findings-acl.466/.

Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 32369–32399. PMLR, 21–27 Jul 2024b. URL https://proceedings.mlr.press/v235/liu24cb.html.

Yong Liu, Xingjian Su, Shiyu Wang, Haoran Zhang, Haixuan Liu, Yuxuan Wang, Zhou Ye, Yang Xiang, Jianmin Wang, and Mingsheng Long. Timer-s1: A billion-scale time series foundation model with serial scaling, 2026. URL https://arxiv.org/abs/2603.04791.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Wenzhe Niu, Zongxia Xie, Yanru Sun, Wei He, Man Xu, and Chao Hao. LangTime: A language-guided unified model for time series forecasting with proximal policy optimization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 46712–46734. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/niu25e.html.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 8748–8763. PMLR, 18–24 Jul 2021. URL https://proceedings.mlr.press/v139/radford21a.html.

Kashif Rasul, Arjun Ashok, Andrew Robert Williams, Hena Ghonia, Rishika Bhagwatkar, Arian Khorasani, Mohammad Javad Darvishi Bayazi, George Adamopoulos, Roland Riachi, Nadhir Hassen, Marin Biloš, Sahil Garg, Anderson Schneider, Nicolas Chapados, Alexandre Drouin, Valentina Zantedeschi, Yuriy Nevmyvaka, and Irina Rish. Lag-llama: Towards foundation models for probabilistic time series forecasting, 2024.

Jun Seo, Hyeokjun Choe, Seohui Bae, Soyeon Park, Jinseok Yang, Dongwan Kang, and Woohyung Lim. Adaptive information routing for multi modal time series forecasting. In *NeurIPS Workshop on Time Series in the Age of Large Models*, 2024. URL https://openreview.net/forum?id=3Im0luRZHS.

ChengAo Shen, Wenchao Yu, Ziming Zhao, Dongjin Song, Wei Cheng, Haifeng Chen, and Jingchao Ni. Multi-modal view enhanced large vision models for long-term time series forecasting. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 93269–93296. Curran Associates, Inc., 2025. doi: 10.52202/085713-3119. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/860c1c657deafe09f64c013c2888bd7b-Paper-Conference.pdf.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023. URL https://arxiv.org/abs/2302.13971.

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 Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025. ISBN 978-1-57735-897-8. doi: 10.1609/aaai.v39i12.33384. URL https://doi.org/10.1609/aaai.v39i12.33384.

Siyuan Wang, Peng Chen, Yihang Wang, Wanghui Qiu, Guo Guo, Bin Yang, and Yang Shu. Unlocking the value of text: Event-driven reasoning and multi-level alignment for time series forecasting. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 31182–31210, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/34a1fc7890141f1ada3d8bc6199cce07-Paper-Conference.pdf.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Xingjian Wu, Jianxin Jin, Wanghui Qiu, Peng Chen, Yang Shu, Bin Yang, and Guo Guo. Aurora: Towards universal generative multimodal time series forecasting. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 103239–103277, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/a8e18adf6489fd3a27417d128660eaa9-Paper-Conference.pdf.

Chang Xu, Dacheng Tao, and Chao Xu. A survey on multi-view learning, 2013. URL https://arxiv.org/abs/1304.5634.

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jianxin Yang, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Xuejing Liu, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zhifang Guo, and Zhihao Fan. Qwen2 technical report, 2024. URL https://arxiv.org/abs/2407.10671.

Luoxiao Yang, Yun Wang, Xinqi Fan, Israel Cohen, Jingdong Chen, and Zijun Zhang. Vitime: Foundation model for time series forecasting powered by vision intelligence, 2025. URL https://arxiv.org/abs/2407.07311.

Yueyang Yao, Jiajun Li, Xingyuan Dai, MengMeng Zhang, Xiaoyan Gong, Fei-Yue Wang, and Yisheng Lv. Context-aware probabilistic modeling with llm for multimodal time series forecasting, 2025. URL https://arxiv.org/abs/2505.10774.

Jing Zhao, Xijiong Xie, Xin Xu, and Shiliang Sun. Multi-view learning overview: Recent progress and new challenges. *Information Fusion*, 38:43–54, 2017. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2017.02.007. URL https://www.sciencedirect.com/science/article/pii/S1566253516302032.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 78478–78497. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/zhong25a.html.

Tian Zhou, Peisong Niu, xue wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained lm. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 43322–43355. Curran Associates, Inc., 2023. doi: 10.52202/075280-1877. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/86c17de05579cde52025f9984e6e2ebb-Paper-Conference.pdf.

A Overview of Supplementary Material

This supplementary material provides additional details, implementation specifics, and extended experimental analyses to support the main manuscript. Below is an overview of the contents included in the appendix:

- **Section B: Detailed Comparison of Context Integration Strategies**
  Provides a comprehensive taxonomy and comparison between RouteCast and existing methods with Table 5.
- **Section C: Dataset Details**
  Provides benchmark dataset statistics, context length/forecast horizon configurations with Table 6 and preprocessing procedures, and dataset descriptions.
- **Section D: Time Series Description**
  Describes the detailed prompts and generation process using Qwen3VL for textual time-series captioning.
- **Section E: RouteCast Computation Flow**
  Presents the step-by-step mathematical workflow of the RouteCast framework with Algorithm 1.
- **Section F: RouteCast Components**
  Elaborates a conceptual comparison between full fine-tuning and RouteCast with Figure 5.
- **Section G: Implementation Details**
  Provides the implementation details for RouteCast.
- **Section H: Baseline Implementation Details**
  Provides the full implementation details for all baseline models evaluated in our experiments.
- **Section I: RouteCast Full Result**
  Presents the full results of RouteCast and the baselines across datasets, including standard deviations over five runs with Table 7.
- **Section J: Effect of Pretrained Backbones and Encoders**
  Details the performance of RouteCast across time-series backbones, vision encoders, and text encoders, including standard deviations over three runs with Table 8.
- **Section K: Context Distillation and Context Routing Ablation**
  Reports comprehensive ablation study results isolating the impact of distillation and routing mechanisms with Table 9.
- **Section L: Implementation Details of Context Distillation and Context Routing**
  Details the variants of Context Distillation and Context Routing with Table 10
- **Section M: Effective Multi-view Representation Strategies**
  Compares multi-view representation pooling strategies with Table 11 and 12 and justifies our selected design choices.
- **Section N: Multi-view Gain Analysis**
  Quantifies the performance improvement provided by incorporating text and visual auxiliary views with Figure 6 and 7
- **Section O: Additional Baselines**
  Presents additional baselines and compares their performance with Table 13.
- **Section P: Long-term Forecast**
  Evaluates model stability and performance across extended forecasting horizons with Table 14.
- **Section Q: Analysis on Context Routing**
  Investigates routing behaviors through attention weight distribution with Figure 8, 9 and 10.
- **Section R: Qualitative Analysis**
  presents sample predictions across diverse domain scenarios to illustrate model capabilities with Figure 11.
- **Section S: Heatmaps**
  Visualizes attention weights across views with Figure 12, 13 and 14.
- **Section T: Parameter Efficiency and Computational Cost**
  Analyzes parameter efficiency and computational cost of RouteCast with Table 15 and 16.

B Detailed Comparison of Context Integration Strategies

Method	TS	V	T	Conditioning	Context Usage	Instance Ctrl	Frozen
Time-LLM	✓		✓	Prompt	Fixed	×	◯
LangTime	✓		✓	Prompt	Fixed	×	×
VisionTS	✓	✓		Direct	Fixed	×	◯
DMMV	✓	✓		Direct	Fixed	Partial	△
Time-VLM	✓	✓	✓	Direct	Coupled	×	◯
TimeCMA	✓		✓	Alignment	Fixed	Partial	△
VoT	✓		✓	Alignment	Fixed	×	×
RouteCast (Ours)	✓	✓	✓	**Routing**	**Adaptive**	**Full**	◯

Table 5: Comparison of context integration strategies in time-series forecasting. TS, V, and T denote time-series, vision, and text modalities, respectively. Instance Ctrl refers to instance-level control over contextual information, and Frozen indicates whether the backbone model remains frozen during training. Prior work relies on static or implicitly coupled context usage, whereas RouteCast enables explicit instance-level routing.

We provide a categorization and structural comparison of existing time-series forecasting approaches in Table 5. Specifically, we position our proposed method, **RouteCast**, relative to prior work along five key dimensions:

- **Supported Views (TS / V / T):** Indicates whether the model integrates time-series (TS), vision (V), or text (T) views.
- **Conditioning Mechanism:** Describes how auxiliary views or contextual information are condition-infused into the forecasting pipeline (e.g., via hard-coded Prompts, Direct transfer of inductive biases, cross-modal Representation Alignment, or dynamic Routing).
- **Context Usage:** Reflects whether the representation of context remains static and invariant to inputs (*Fixed*), entangled across views (*Coupled*), or dynamically adjusted per sample (*Adaptive*).
- **Instance-Level Control (Instance Ctrl):** Indicates whether the method allows fine-grained, explicit regulation of each view's contribution on a per-instance basis.
- **Frozen Backbone:** Specifies whether the underlying pretrained foundation backbones are kept frozen.

Key Observations and Positioning. As summarized in Table 5, prior context integration strategies suffer from rigid, input-invariant representations, entangled multi-view interactions, or loss of pretrained generalization caused by updating backbone parameters during training. In contrast, **RouteCast** overcomes these structural trade-offs through *instance-dependent context routing*. By decoupling context inference from utilization while keeping the pretrained backbones strictly frozen, RouteCast enables dynamic, parameter-efficient, and instance-level control across time-series, vision, and text views.

C Dataset Details

To evaluate the effectiveness of our **RouteCast**, we adopt a subset of benchmarks used in the zero-shot evaluation of Chronos (Ansari et al., 2024). For a fair comparison, we follow the same settings as Chronos, using identical context lengths C and forecast horizons H. Summary statistics including forecasting configuration for all datasets are presented in Table 6. Each time series is split into training, validation, and test sets using a 60:20:20 ratio based on its sequence length. In general, we apply standardization within each input-target window pair. However, due to dataset-specific properties, such as short sequence lengths or constant segments, we adopt alternative preprocessing strategies where necessary, as detailed below. ***1) COVID-19 Deaths***[1] contains 266 daily time series representing COVID-19 death counts across various countries and states, from 22/01/2020 to 20/08/2020. Each series consists of only 212 time steps, which is relatively short given our context and forecast window sizes. To ensure proper sample diversity, we shuffle the time series before applying the 6:2:2 split. Since many context windows contain constant values, standardization is

Table 6: Summary statistics of Dataset.

Dataset	Domain	Frequency	# of Series	Series Length			C	H
				min	avg	max		
NN5 Daily	Finance (cash withdrawal volumes)	1 Day	111	-	791	-	56	56
Australian Electricity	Energy (electricity demand)	30 Minutes	5	230736	231052	232272	48	48
Tourism Monthly	Others (tourism demand)	1 Month	366	91	298	333	24	24
COVID-19 Deaths	Health (death across countries)	1 Day	266	-	212	-	30	30
Car Parts	Retail (car component sales)	1 Month	2674	-	51	-	12	12
CIF 2016	Banking (real-time banking demand)	1 Month	72	28	98	120	12	12
Dominick	Retail (stock-keeping units)	1 Week	100014	201	296	399	8	8
Hospital	Health (patient count)	1 Month	767	-	84	-	12	12
ETTh	Energy (transformer temperature)	1 Hour	14	-	17420	-	24	24
ETTm	Energy (transformer temperature)	15 Minutes	14	-	69680	-	96	24

C Context Length. H Forecast Length.

applied over the entire series to avoid division errors. ***2) NN5 Daily***[2] includes 111 daily time series from the UK banking sector, capturing ATM cash withdrawal volumes. We adopt the preprocessing strategy mentioned earlier. ***3) Car Parts***[1] comprises monthly sales data for various car components between January 1998 and March 2002. Due to the limited sequence lengths, we apply series-level shuffling followed by a 6:2:2 split. Additionally, to prevent numerical instability caused by constant windows, standardization is performed over entire sequences. ***4) Australian Electricity***[1] consists of half-hourly electricity demand data from five Australian states. Each series contains over 200,000 time steps. To reduce computational overhead while preserving temporal diversity, we use only the first 15,000 steps per series for training and evaluation. ***5) CIF 2016***[1] originates from the CIF 2016 forecasting competition and includes 72 monthly banking-related time series: 24 real and 48 synthetic. Due to their short lengths, we apply the same shuffle-and-split strategy and use series-level standardization. ***6) Dominick***[1] contains over 100,000 weekly time series measuring the profitability of individual stock-keeping units in a retail setting. For computational feasibility, we randomly sample 100 series after shuffling and apply standard preprocessing. ***7) Hospital***[1] includes monthly time series reflecting patient counts related to medical products, recorded from January 2000 to December 2006. Due to the limited sequence lengths, we follow the same preprocessing approach as with Car Parts and CIF 2016. ***8) Tourism Monthly***[1] is derived from the Kaggle Tourism Forecasting competition and includes a collection of short monthly tourism-related time series. As with similar datasets, we shuffle and split the series and apply standardization at the full-series level. ***9) ETT***[3] contains oil temperature measurements and six types of external power load features collected from two electricity transformers in China over a two-year period. The dataset has four variants: ETTh1, ETTh2, ETTm1, and ETTm2, corresponding to hourly ('h') and 15-minute ('m') sampling frequencies. In our experiments, we use only the 'OT' series, which represents the target oil temperature.

D Time Series Description

To provide textual information corresponding to the input time series to the text encoder, we generate a natural-language description for each input window. Specifically, we leverage the *Qwen/Qwen3-VL-8B-Instruct* via Huggingface to produce a *time series description* that summarizes the temporal characteristics of the input.

For each input window, the raw time series values are provided to the model along with a fixed prompt that instructs the model to analyze the data and describe its temporal structure. The prompt explicitly asks the model to focus on two key aspects: (i) the overall trend, including its direction, shape, and consistency, and (ii) the presence of seasonality or repeating patterns. If the evidence for either trend or seasonality is weak or insufficient, the model is instructed to explicitly state this uncertainty.

[1]We obtained those datasets from https://huggingface.co/datasets/autogluon/chronos_datasets

[2]We obtained NN5 dataset from https://zenodo.org/records/4656117 due NN5 dataset in HuggingFace Chronos dataset repository has missing values.

[3]We obtained ETT dataset from https://github.com/zhouhaoyi/ETDataset.

The prompt used to generate the time series descriptions is shown below:

Prompt for time series description

You are given a sequence of time series values.
The time series data is:
{time series}

Analyze the time series and focus primarily on identifying:
- the overall trend (direction, shape, and consistency)
- any evidence of seasonality or repeating patterns

Return your answer strictly in valid JSON format and do not include any extra text.

The JSON must contain only one field:
- "description": a concise and clear natural-language description of the trend and seasonality observed in the time series. If seasonality is weak or not detectable, explicitly state that.

If the data is insufficient to confidently determine trend or seasonality, clearly express this uncertainty in the description.

The generated description is then used as the textual input corresponding to the same time series window. This approach allows the text encoder to access high-level semantic information about the temporal dynamics of the input without relying solely on raw numerical values.

E RouteCast Computation Flow

Algorithm 1 summarizes the complete computation flow of RouteCast. Starting from an observed time-series history X, visual and textual views are derived from the same input and represented together with the numerical view. Context Distillation first summarizes these representations and generates instance-conditioned soft prompts for the frozen vision and text encoders. The resulting auxiliary representations are then projected into the TSFM feature space and used to construct an instance-specific TSFM prompt.

Context Routing subsequently regulates the relative influence of the distilled visual and textual representations on the temporal hidden states. At each routing step, RouteCast computes compatibility scores between the current temporal representation and each auxiliary view, normalizes these scores across views, and forms a weighted auxiliary representation that is added through a residual update. Because routing weights are recomputed after each frozen TSFM block, view utilization can vary across forecasting instances, temporal positions, and representation depth. The pretrained vision encoder, text encoder, and TSFM backbone remain frozen throughout training.

F RouteCast Components

Figure 5 presents a conceptual comparison between conventional full fine-tuning (FT) and the proposed RouteCast framework. While FT relies solely on the raw time series input for downstream forecasting, RouteCast transforms the time series into complementary vision and text representations to provide additional contextual information for the TSFM. As illustrated in Figure 5b, Context Distillation (CD) extracts contextual information from multiple view and generates instance-specific soft prompts that enable the TSFM to utilize task-relevant multi-view signals. However, the contextual embeddings from each view are directly injected into the TSFM transformer layers, making it difficult to explicitly control the contribution of individual views. Figure 5c further introduces Context Routing (CR) together with CD, where contextual embeddings from each view are not directly propagated into the TSFM. Instead, CR selectively emphasizes or suppresses information from each view through cross-attention-based routing, and the contributions are iteratively refined across TSFM transformer layers in an input-dependent manner. Together, CD and CR enable RouteCast to achieve adaptive and efficient multi-view forecasting by conditioning the TSFM on contextual relevance.

Algorithm 1 RouteCast Computation Flow

Require: Observed time-series history $X \in \mathbb{R}^{C \times D}$, derived visual view $V = g_v(X)$, derived textual view $T = g_t(X)$, number of TSFM layers N

Context Distillation

Input Representation

$O_v = \mathrm{Embed}_v(V)$

$O_t = \mathrm{Embed}_t(T)$

$O_x = \mathrm{Patch}(X)$

Instance-Conditioned Auxiliary Prompt Generation

$\bar{O}_m = \mathrm{MeanPool}(O_m), \quad m \in \{v, t, x\}$

$$\mathrm{Proj}_{i \to m}(\bar{O}_i) = \begin{cases} \bar{O}_i, & i = m, \\ P_{i \to m}(\bar{O}_i), & i \neq m, \end{cases} \quad i, m \in \{v, t, x\}$$

for $m \in \{v, t\}$ **do**

$$c_m = G_m \left(\underset{i \in \{v,t,x\}}{\mathrm{Concat}} \left[\mathrm{Proj}_{i \to m}(\bar{O}_i) \right] \right)$$

end for

Instance-Conditioned View Refinement

$E_v = \mathrm{FirstToken}\left(F_v([c_v; O_v])\right)$

$E_t = \mathrm{MeanPool}\left(F_t([c_t; O_t])\right)$ ▷ F_v, F_t are frozen

$z_v = P_{v \to x}(E_v)$

$z_t = P_{t \to x}(E_t)$

TSFM Prompt Generation

$p_x = G_x\left(\mathrm{Concat}[z_v; z_t; \bar{O}_x]\right)$

Context Routing

Initialize Multi-View Context

$\mathcal{M} = \{v, t\}$

$Z = \{z_m \mid m \in \mathcal{M}\}$

Initial View Routing

for each time-series position j **do**

$$s_{j,m}^{(0)} = \frac{(W_q O_{x,j})^\top (W_k z_m)}{\sqrt{d}}, \quad \alpha_{j,m}^{(0)} = \frac{\exp(s_{j,m}^{(0)})}{\sum_{u \in \mathcal{M}} \exp(s_{j,u}^{(0)})}, \quad \forall m \in \mathcal{M}$$

$r_j^{(0)} = \sum_{m \in \mathcal{M}} \alpha_{j,m}^{(0)} W_{\mathrm{val}} z_m$

$\tilde{O}_{x,j} = O_{x,j} + r_j^{(0)}$

end for

$H^{(0)} = [p_x; \tilde{O}_x]$

Iterative View Routing through the Frozen TSFM

for $\ell = 0, 1, \ldots, N-1$ **do**

$H^{(\ell+\frac{1}{2})} = \mathrm{TSFMBlock}_{\ell+1}\left(H^{(\ell)}\right)$ ▷ frozen

for each temporal position j **do**

$$s_{j,m}^{(\ell+1)} = \frac{(W_q h_j^{(\ell+\frac{1}{2})})^\top (W_k z_m)}{\sqrt{d}}, \quad \alpha_{j,m}^{(\ell+1)} = \frac{\exp(s_{j,m}^{(\ell+1)})}{\sum_{u \in \mathcal{M}} \exp(s_{j,u}^{(\ell+1)})}, \quad \forall m \in \mathcal{M}$$

$r_j^{(\ell+1)} = \sum_{m \in \mathcal{M}} \alpha_{j,m}^{(\ell+1)} W_{\mathrm{val}} z_m$

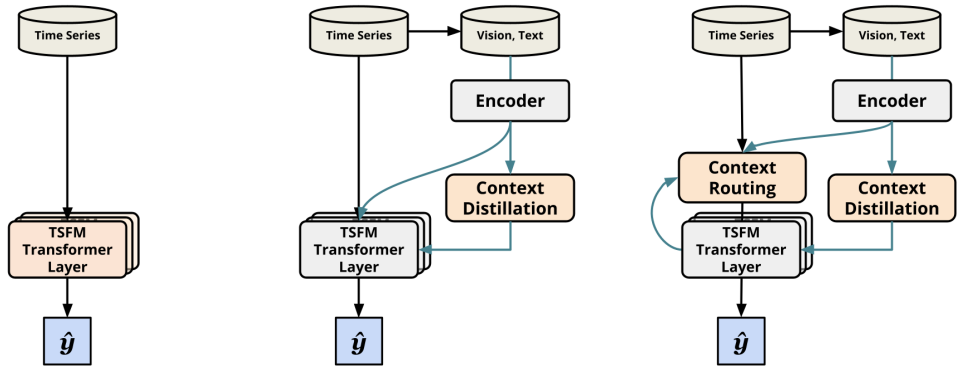
$h_j^{(\ell+1)} = h_j^{(\ell+\frac{1}{2})} + r_j^{(\ell+1)}$

end for

end for

Forecasting

$\hat{Y} = f_{\mathrm{head}}\left(H^{(N)}\right)$



(a) Full Fine-tuning (b) Context Distillation (c) CD + CR

Figure 5: Conceptual visualization of RouteCast components, Context Distillation (CD) and Context Routing (CR) compared to Full Fine-tuning.

G Implementation Details

RouteCast operates over numerical, visual, and textual views derived from the same observed time-series history. For the visual view, we use CLIP (Radford et al., 2021) and BLIP (Li et al., 2022) as alternative frozen vision encoders. For the textual view, we use Qwen2 (Yang et al., 2024) and Llama (Touvron et al., 2023) as alternative frozen text encoders. For the numerical view, we instantiate RouteCast with Chronos (Ansari et al., 2024), Timer (Liu et al., 2024b), and TimesFM (Das et al., 2024) as alternative frozen TSFM backbones. The pretrained checkpoints used in our experiments are: **Vision**: *openai/clip-vit-base-patch32* and *Salesforce/blip-image-captioning-base*; **Text**: *Qwen/Qwen2-1.5B-Instruct* and *huggyllama/llama-7b*; and **Time Series**: *amazon/chronos-bolt-base*, *thuml/timer-base-84m*, and *google/timesfm-2.5-200m-pytorch*. Both auxiliary views are constructed solely from the observed input window. For the visual view, we transform the numerical history into a plotted time-series representation following ViTST (Li et al., 2023), implemented using *matplotlib*. For the textual view, we use *Qwen/Qwen3-VL-8B-Instruct* with a fixed generation prompt to produce a concise description of the observed time-series window, focusing on properties such as trend and seasonality. No future observations or independent exogenous information are used to construct either auxiliary view. The complete text-generation prompt and further details are provided in Appendix D. All trainable components are optimized using AdamW with a learning rate of 2×10^{-5} for 10 epochs. All trainable experiments are repeated using five random seeds, and we report the mean and standard deviation across runs. Experiments are implemented in PyTorch and executed on a single NVIDIA H100 NVL GPU with 94 GB memory. For each run, we select the checkpoint with the lowest validation MSE and report its performance on the held-out test set.

H Baseline Implementation Details

In this section, we provide the full implementation details for all baseline models evaluated in our experiments, categorized into Time-Series Foundation Models (TSFMs) and controlled multi-view baselines.

Time-Series Foundation Models (TSFMs). We evaluate both Zero-Shot (ZS) and Fine-Tuned (FT) variants of prominent TSFMs.

Zero-Shot (ZS) Baselines. All pretrained zero-shot TSFM baselines are obtained from their official public repositories hosted on Hugging Face. The specific model checkpoints used for our experiments are as follows: **Chronos:** *amazon/chronos-bolt-base*, **Chronos-2:** *amazon/chronos-2*, **Timer:** *thuml/timer-base-84m*, **Timer-S1:** *bytedance-research/Timer-S1*, **TimesFM:** *google/timesfm-2.5-200m-pytorch*, **MOMENT:** *AutonLab/MOMENT-1-base*, **MOIRAI:** *Salesforce/moirai-2.0-R-small*, and **TTM:** *ibm-granite/granite-timeseries-ttm-r2*. For all zero-shot evaluations, we adhere strictly to the inference pipelines provided by the respective authors.

Fine-Tuned (FT) Baselines. To establish upper-bound performance baselines for the backbones used within **RouteCast**, we additionally evaluate fully fine-tuned (FT) versions of Chronos, Timer, and TimesFM. All FT models are trained end-to-end using Mean Squared Error (MSE) loss on the target numerical time-series data. The models are fine-tuned for 10 epochs using the Adam optimizer with an initial learning rate of 1×10^{-3}. During training, we monitor the validation loss at the end of each epoch and save the best checkpoint, which is subsequently used to report final test performance.

Controlled Multi-View Baselines. To dissect whether RouteCast's performance gains stem merely from incorporating auxiliary visual and textual modalities, we implement a 2×2 factorial family of controlled multi-view baselines: $\{\text{MLP}, \text{Transformer}\} \times \{\text{scratch}, \text{pretrained}\}$. All multi-view baselines receive identical numerical, visual, and textual inputs, utilizing the exact same preprocessing logic as RouteCast.

Auxiliary Encoders. For the vision encoder, the pretrained variant uses the pretrained CLIP vision encoder matching the configuration in RouteCast. The from-scratch variant (scratch) uses a standard Transformer encoder architecture trained from scratch, where input images are preprocessed identically to CLIP, a learnable `[CLS]` token is prepended to the patch sequence, and its final hidden state is extracted as the vision embedding. This encoder consists of 4 Transformer layers, 4 attention heads, and a hidden dimension (d_{model}) of 768. For the text encoder, the pretrained variant uses the pretrained Qwen model matching RouteCast. The from-scratch variant (scratch) employs a Transformer encoder initialized from scratch and tokenized using the Qwen tokenizer. Because this architecture follows a decoder-style setup, causal masks are applied during self-attention, and average pooling across sequence tokens is applied to obtain the final text embedding following RouteCast's strategy. The architecture consists of 4 layers, 12 attention heads, and a hidden dimension (d_{model}) of 768.

Integration & Forecasting Architectures. For MLP-based integration ($\text{MLP}_{\text{scratch}}$ / $\text{MLP}_{\text{pretrained}}$), raw numerical time-series inputs of shape $(C,)$ are directly mapped to a D-dimensional representation using a linear/MLP projection without patching. The numerical representation, vision embedding, and text embedding are concatenated along the feature dimension and passed through a multi-layer perceptron with a hidden dimension of 256 to project directly to the target forecasting horizon. For Transformer-based integration ($\text{Transformer}_{\text{scratch}}$ / $\text{Transformer}_{\text{pretrained}}$), the architecture follows an Encoder-Decoder Transformer modeled after Chronos. The numerical time-series is partitioned into non-overlapping patches of length 16 and embedded into a token sequence following Chronos. The extracted vision and text embeddings are then concatenated with the embedded temporal patch tokens and fed into the Transformer model to generate multi-view integrated forecasts.

Collectively, this controlled suite allows us to systematically verify whether the performance gains of RouteCast originate from the pretrained time-series backbone, the pretrained auxiliary representations, the overall model capacity, or the multi-view integration mechanism itself.

I ROUTECAST FULL RESULT

Table 7: Full results of RouteCast and baselines with standard deviations over five runs. **Bold** and underlined indicate the best and second-best results for each dataset, respectively.

	NN5	Aus	Tour	COV	Car	CIF	Dom	Hos	ETTh1	ETTh2	ETTm1	ETTm2
Chronos ZS	0.791	2.670	2.468	0.156	1.303	7.461	1.340	2.613	0.289	0.172	0.128	0.083
Chronos FT	0.481 ± 0.018	0.385 ± 0.050	1.367 ± 0.017	0.127 ± 0.010	0.997 ± 0.002	7.488 ± 0.162	**0.952 ± 0.004**	2.364 ± 0.022	0.273 ± 0.013	0.144 ± 0.004	0.106 ± 0.004	0.037 ± 0.001
Timer ZS	1.099	1.462	2.622	0.218	0.957	7.597	1.194	2.455	0.315	0.282	0.139	0.084
Timer FT	1.097 ± 0.002	1.539 ± 0.000	2.466 ± 0.009	0.385 ± 0.002	0.919 ± 0.000	7.727 ± 0.001	2.148 ± 0.005	2.470 ± 0.002	0.672 ± 0.000	0.647 ± 0.001	0.241 ± 0.001	0.220 ± 0.000
TimesFM ZS	1.759	2.282	3.170	1.208	1.073	8.075	3.733	2.634	1.895	1.896	1.998	2.015
TimesFM FT	0.847 ± 0.281	0.476 ± 0.087	1.634 ± 0.311	0.343 ± 0.238	3.370 ± 5.113	7.520 ± 0.305	2.408 ± 2.261	2.526 ± 0.233	1.949 ± 3.190	0.436 ± 0.413	0.239 ± 0.266	0.193 ± 0.160
MOMENT ZS	1.130	1.576	2.690	0.517	1.052	7.748	2.026	2.562	0.368	0.275	0.245	0.222
MOIRAI ZS	0.529	1.199	2.596	0.189	1.046	7.498	1.057	2.574	0.307	0.243	0.122	0.063
TTM ZS	1.117	1.512	2.602	0.176	0.991	7.384	1.074	2.448	0.355	0.358	0.118	0.121
Chronos-2 ZS	0.519	2.291	2.241	0.132	1.132	7.411	1.206	2.621	0.290	0.181	0.126	0.088
Timer-S1 ZS	0.525	1.149	2.239	0.250	1.061	7.568	1.005	2.474	0.337	0.169	0.124	0.077
$\text{MLP}_{\text{scratch}}$	0.481 ± 0.001	0.430 ± 0.007	1.355 ± 0.002	0.072 ± 0.002	**0.908 ± 0.000**	7.394 ± 0.026	1.219 ± 0.016	**2.317 ± 0.001**	0.278 ± 0.002	0.148 ± 0.004	0.102 ± 0.001	0.042 ± 0.002
$\text{Transformer}_{\text{scratch}}$	0.482 ± 0.003	0.429 ± 0.006	1.355 ± 0.003	0.072 ± 0.002	**0.908 ± 0.000**	7.392 ± 0.025	1.219 ± 0.024	2.318 ± 0.002	0.278 ± 0.002	0.149 ± 0.003	0.102 ± 0.000	0.041 ± 0.001
$\text{MLP}_{\text{pretrained}}$	0.481 ± 0.001	0.432 ± 0.007	**1.354 ± 0.003**	**0.071 ± 0.004**	**0.908 ± 0.001**	7.392 ± 0.023	1.220 ± 0.024	2.318 ± 0.002	0.278 ± 0.001	0.149 ± 0.004	0.102 ± 0.001	0.041 ± 0.002
$\text{Transformer}_{\text{pretrained}}$	0.480 ± 0.001	0.431 ± 0.005	1.355 ± 0.003	0.072 ± 0.002	**0.908 ± 0.000**	7.394 ± 0.025	1.210 ± 0.012	**2.317 ± 0.002**	0.278 ± 0.002	0.149 ± 0.003	0.102 ± 0.001	0.041 ± 0.001
RouteCast(Ours)	**0.465 ± 0.006**	**0.332 ± 0.015**	1.359 ± 0.002	0.107 ± 0.006	0.951 ± 0.004	**7.329 ± 0.033**	0.971 ± 0.027	2.322 ± 0.004	**0.266 ± 0.003**	**0.136 ± 0.002**	**0.095 ± 0.001**	**0.032 ± 0.002**

Table 7 presents the complete evaluation results across all datasets and baselines, reporting performance metrics averaged over five independent runs with distinct random seeds, along with their corresponding standard deviations. RouteCast consistently outperforms all baselines across the majority of datasets. These comprehensive results, backed by clear statistical robustness, further confirm the crucial roles of input-dependent conditioning via Context Distillation and iterative modulation via Context Routing in driving RouteCast's superior performance.

J Effect of Pretrained Backbones and Encoders

Table 8: Complete results of RouteCast across various backbone and encoder variants (mean ± std over three runs for RouteCast variants, five runs for Full Fine-tuning (FT) variants).

TS	VE	TE	NN5	Aus	Tour
Chronos	ZS		0.791	2.670	2.468
	FT		0.481 ± 0.018	0.385 ± 0.050	1.367 ± 0.017
	CLIP	×	0.470 ± 0.007	0.335 ± 0.001	1.364 ± 0.002
	BLIP	×	0.469 ± 0.006	0.369 ± 0.028	1.362 ± 0.006
	×	Qwen	0.706 ± 0.015	0.421 ± 0.038	1.406 ± 0.009
	×	Llama	0.541 ± 0.114	0.401 ± 0.010	1.365 ± 0.002
	CLIP	Qwen	0.465 ± 0.006	0.332 ± 0.015	1.359 ± 0.002
	CLIP	Llama	0.464 ± 0.006	0.325 ± 0.008	1.361 ± 0.005
	BLIP	Qwen	0.463 ± 0.002	0.326 ± 0.013	1.354 ± 0.005
	BLIP	Llama	**0.459 ± 0.003**	**0.322 ± 0.007**	**1.352 ± 0.006**
Timer	ZS		1.099	1.462	2.622
	FT		1.097 ± 0.002	1.539 ± 0.000	2.466 ± 0.009
	CLIP	Qwen	**0.496 ± 0.011**	**0.291 ± 0.002**	**1.345 ± 0.002**
TimesFM	ZS		1.759	2.282	3.170
	FT		0.847 ± 0.281	0.476 ± 0.087	1.634 ± 0.311
	CLIP	Qwen	**0.470 ± 0.008**	**0.300 ± 0.007**	**1.343 ± 0.003**

Table 8 presents the complete evaluation results across various combinations of time-series backbones, vision encoders, and text encoders, averaged over three independent runs with distinct random seeds alongside their standard deviations. RouteCast consistently outperforms both zero-shot and fine-tuning baselines across datasets regardless of the backbone or encoder paired. These results demonstrate the robust efficacy of Context Distillation and Context Routing, confirming that RouteCast's performance gains are truly backbone- and encoder-agnostic.

K Context Distillation and Context Routing Ablation

Table 9: Impact of Context Distillation (CD) and Context Routing (CR) (mean ± std over three runs for RouteCast variants, five runs for Full Fine-tuning (FT)).

CD	CR	NN5	Aus	Tour	Avg
ZS		0.791	2.670	2.468	1.976
FT		0.481 ± 0.018	0.385 ± 0.050	1.367 ± 0.017	0.744
×	×	0.473 ± 0.005	0.391 ± 0.010	1.401 ± 0.002	0.755
◯	×	0.474 ± 0.004	0.375 ± 0.002	1.366 ± 0.003	0.738
×	◯	0.468 ± 0.004	0.353 ± 0.010	1.388 ± 0.009	0.736
◯	◯	**0.465 ± 0.006**	**0.332 ± 0.015**	**1.359 ± 0.002**	**0.719**

This section provides the full results of the ablation studies on Context Distillation (CD) and Context Routing (CR). We report performance across multiple datasets for different implementation choices, including static versus CD and heuristic versus CR, averaged over three independent runs with distinct random seeds alongside their standard deviations.

While the main text presents only a summary of these results using bar plots for clarity, Table 9 shows the complete numerical results, allowing a detailed comparison of each design choice and its effect on RouteCast performance.

L Implementation Details of Context Distillation and Context Routing

Note that the experiments in this section are single-run explorations conducted to finalize the module implementations before main benchmarking; thus, slight numerical variations may exist across tables under identical baseline settings.

Context Distillation. To investigate whether explicitly modeling interactions among multi-view inputs is necessary when constructing CD, we compare two implementations of CD, an MLP-based generator and a Transformer-based generator. As shown in Table 10, the Transformer-based implementation consistently outperforms the MLP-based variant, indicating that capturing dependencies across views during prompt generation is crucial for effective conditional prompting. This suggests that CD benefits from architectures that can model structured interactions across views, rather than relying on simple aggregation mechanisms. Based on these results, we adopt the Transformer-based CD as the default design in RouteCast.

Context Routing. We evaluate two CR strategies: a Layerwise routing scheme, where interactions across views are modeled at each transformer layer, and a Shared routing scheme, where a single routing module is shared across layers. While the Layerwise CR achieves better performance, it introduces additional parameters and computational overhead. In contrast, the Shared CR provides competitive performance with significantly improved parameter efficiency. Considering RouteCast's goal of a parameter-efficient multi-view forecasting framework, we adopt Shared CR as the default configuration to balance performance and parameter efficiency.

Table 10: Implementation of Context Distillation (CD) and Context Routing (CR). CD compares Transformer (T) and MLP (M), while CR compares Shared (S) and Layerwise (L).

CD	CR	NN5	Aus	Tour	Avg
ZS		0.791	2.670	2.468	1.977
FT		0.464	0.361	1.390	0.738
T	S	0.464	0.343	**1.357**	0.721
M	S	0.466	0.334	1.379	0.726
T	L	**0.463**	**0.310**	1.373	**0.715**

M Effective Multi-view Representation Strategies

Note that the experiments in this section are single-run explorations conducted to finalize the module implementations before main benchmarking; thus, slight numerical variations may exist across tables under identical baseline settings.

To analyze the effect of embedding extraction strategies on RouteCast performance, we compare different pooling methods for Vision and Text views. For Vision backbones (CLIP, BLIP), we compare using the First Token versus Average Pooling, while for Text backbones (Qwen, Llama), we compare using the Last Token versus Average Pooling.

Table 11: Comparison of different Vision embedding extraction strategies for RouteCast. We compare using the First Token versus Average Pooling from CLIP and BLIP, both with and without the Qwen text model. For each dataset and model configuration, the best-performing strategy is highlighted in bold.

VE	TE	First Token				Average Pooling			
		NN5	Aus	Tour	Avg	NN5	Aus	Tour	Avg
CLIP	×	**0.4658**	**0.3450**	**1.3535**	**0.7214**	0.4735	0.3760	1.3808	0.7434
BLIP	×	**0.4625**	**0.3517**	**1.3559**	**0.7234**	0.4717	0.3787	1.3750	0.7418
CLIP	Qwen	**0.4635**	0.3431	**1.3573**	**0.7213**	0.4671	**0.3334**	1.3720	0.7242
BLIP	Qwen	**0.4601**	0.3320	**1.3499**	**0.7140**	0.4641	**0.3171**	1.3764	0.7192

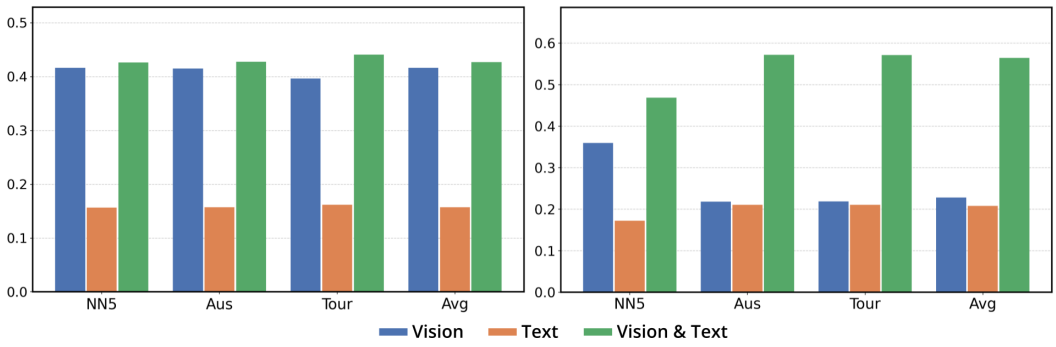


Figure 6: Comparison of Multi-view Gain (MVG) and Per-view Gain across datasets. Left: MVG computed between uni-view and multi-view forecasting settings. Right: Per-view Gain obtained by removing each view from the full multi-view model. Results are reported using a CLIP vision encoder and a Qwen text encoder. For visualization, model outputs were first scaled using min-max normalization and then passed through a softmax function within each dataset.

Vision. For vision encoders, using the First Token consistently outperforms Average Pooling across both CLIP and BLIP backbones as shown in Table 11. This result aligns with the design of vision transformer–based encoders, where the first token serves as a global representation that aggregates information from all image patches through self-attention. During pretraining, both CLIP and BLIP optimize this token to capture high-level semantic information for cross-modal alignment, making it particularly suitable as a compact representation for multi-view conditioning. In contrast, Average Pooling treats all patch tokens equally, which may dilute salient visual features that are most relevant for forecasting. These findings suggest that leveraging the pretrained global representation token is more effective than uniform spatial aggregation for the visual view. Based on this observation, we adopt the First Token as the default embedding extraction strategy for Vision in RouteCast.

Table 12: Comparison of different Text embedding extraction strategies for RouteCast. We compare using the Last Token versus Average Pooling from Qwen and Llama, both with and without the CLIP vision model. For each dataset and model configuration, the best-performing strategy is highlighted in bold.

VE	TE	Last Token				Average Pooling			
		NN5	Aus	Tour	Avg	NN5	Aus	Tour	Avg
×	Qwen	**0.6556**	0.5786	1.4074	0.8805	0.6680	**0.3753**	**1.4060**	**0.8164**
×	Llama	0.6662	0.3882	**1.3567**	0.8037	**0.4706**	**0.3823**	1.3680	**0.7403**
CLIP	Qwen	**0.4624**	**0.3304**	1.3604	**0.7177**	0.4635	0.3431	**1.3573**	0.7213
CLIP	Llama	0.4597	0.3112	**1.3499**	0.7069	**0.4558**	**0.3104**	1.3534	**0.7065**

Text. Table 12 reports that Average Pooling consistently outperforms using the Last Token representation when decoder-only language models (Qwen and Llama) are used as text backbones. This behavior can be attributed to the architectural characteristics of decoder-only transformers, where the final token representation is optimized for next-token prediction during pretraining. As a result, the last token embedding may reflect generation-oriented context rather than a balanced summary of the entire textual input. In contrast, Average Pooling aggregates information across all tokens, producing a more stable sequence-level semantic representation that better captures forecasting-relevant context. This suggests that, when decoder-only language models are used for multi-view conditioning rather than text generation, global token aggregation is more suitable than relying on the final token representation. Based on this observation, we adopt Average Pooling as the default embedding extraction strategy for the textual view in RouteCast.

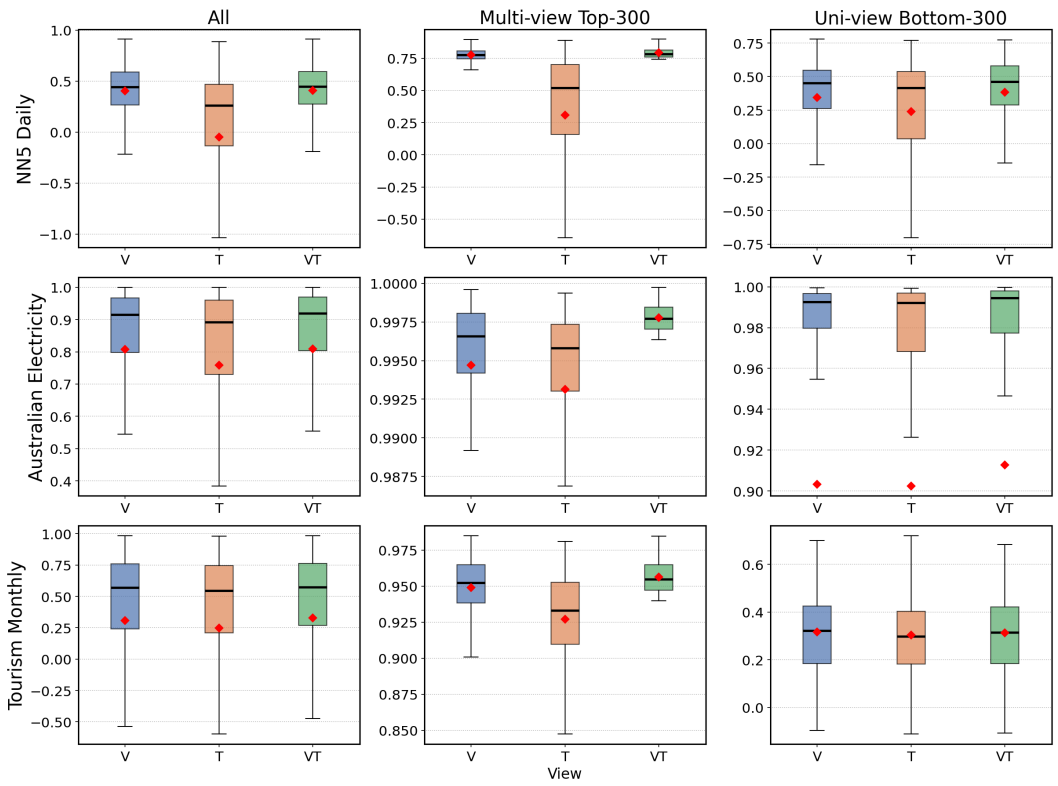


Figure 7: Distribution of instance-level Multi-view Gain across datasets and evaluation cases. Columns correspond to all test instances, top-300 best-predicted instances under the multi-view(VT) setting, and bottom-300 worst-predicted instances under the uni-view setting. Rows represent different datasets. Results are obtained using a BLIP vision encoder and a Llama text encoder. Outliers are omitted for clarity. Black lines indicate medians and red dots indicate means.

N Multi-view Gain Analysis

To quantitatively analyze how multi-view information contributes to forecasting performance, we introduce three complementary metrics. All experiments are conducted using a BLIP-based vision encoder and a Llama-based text encoder. These metrics are designed to evaluate multi-view benefits at both the dataset level and the instance level, where the latter provides a finer-grained analysis of dataset-level observations.

Multi-view Gain (MVG). We define Multi-view Gain (MVG) to measure the relative error reduction achieved by incorporating multiple views compared to uni-view forecasting:

$$MVG = \frac{E_{uni} - E_{multi}}{E_{uni}},$$

where E_{uni} denotes the forecasting error of the uni-view model, and E_{multi} denotes the error of the multi-view model.

Per-view Gain (MVG_v). To quantify the contribution of each view within the full multi-view setting, we define Per-view Gain as:

$$MVG_v = \frac{E_{w/o\,v} - E_{base}}{E_{base}},$$

where $E_{w/o\,v}$ is the error after removing view v, and E_{base} is the error of the full multi-view model. This metric captures the marginal contribution of each view.

Instance-level MVG. To further analyze the distribution of multi-view benefits, we compute instance-level MVG as:

$$MVG(i) = \frac{E_{uni}(i) - E_{multi}(i)}{E_{uni}(i)},$$

where $E_{uni}(i)$ and $E_{multi}(i)$ denote the uni-view and multi-view errors for instance i, respectively. This metric enables fine-grained analysis of how multi-view fusion affects individual forecasting cases.

Figure 6 summarizes Multi-view Gain and Per-view Gain. Across datasets, we observe that multi-view inputs consistently improve forecasting performance over uni-view settings. Furthermore, when comparing individual views, vision generally contributes more to performance improvement than text, indicating that visual context provides stronger predictive signals in our setting.

To further understand how these gains are distributed across samples, Figure 7 presents instance-level MVG distributions. We evaluate all test instances, as well as subsets of top-performing and worst-performing cases under different settings. The results show that the relative ordering of Vision & Text, Vision, and Text remains consistent across datasets and evaluation subsets, indicating that multi-view benefits are broadly stable across instance difficulty levels, while still exhibiting variation across samples.

O Additional Baselines

Table 13: Additional Baselines. **Bold** indicates the best result for each dataset.

	NN5	Aus	Tour	COV	Car	CIF	Dom	Hos	ETT				Avg
									h1	h2	m1	m2	
PatchTST	0.478	0.393	1.423	0.123	0.984	7.961	1.534	**2.311**	0.268	0.138	0.097	0.035	1.312
GPT4TS	0.486	0.439	1.429	0.170	0.995	8.006	1.530	2.325	0.279	0.153	0.103	0.044	1.330
Time-LLM	0.540	0.609	2.071	0.220	1.047	8.102	1.545	2.445	0.282	0.146	0.106	0.049	1.430
Lag-Llama	1.273	1.540	3.153	0.305	1.013	8.392	1.949	2.483	0.302	0.269	0.296	0.376	1.779
RouteCast(Ours)	**0.465**	**0.332**	**1.359**	**0.107**	**0.951**	**7.329**	**0.971**	2.322	**0.266**	**0.136**	**0.095**	**0.032**	**1.197**

In addition to the main comparisons, we include further results with Transformer- and LLM-based models, PatchTST(Nie et al., 2023), GPT4TS(Zhou et al., 2023), Time-LLM(Jin et al., 2024), Lag-Llama(Rasul et al., 2024) in the appendix to provide a broader empirical perspective. We intentionally exclude these models from the main results table to preserve a fair and controlled comparison setting. Specifically, our primary focus is on evaluating improvements within the TSFM paradigm, where models share similar pretraining objectives, architectural inductive biases, and scaling regimes. In contrast, Transformer- and LLM-based approaches differ fundamentally along these dimensions, making direct comparisons less controlled and potentially confounding. Therefore, we present these additional baselines in the appendix, allowing for extended comparison while maintaining clarity and fairness in the main evaluation.

The results in Table 13 further demonstrate the effectiveness and robustness of RouteCast across a broader range of baseline paradigms and model generations. RouteCast consistently delivers superior overall performance across most benchmarks. In particular, RouteCast outperforms the additional baselines by a substantial margin in the majority of cases, indicating that the observed gains are not limited to comparisons within a narrow model family, but generalize across diverse forecasting paradigms and stronger recent variants. These results further validate the effectiveness of our framework in leveraging TSFM representations for accurate and robust time series forecasting.

P Long-term Forecast

In the main experiments, we focus on short-term forecasting settings using evaluation subsets that were not included in the pretraining data of Chronos to strictly avoid potential data leakage. However, to ensure broader comparability with prior literature and to assess the generality of our framework, we additionally report results under commonly adopted long-term forecasting settings. Following standard practice, we use a context length of 512 and forecast horizons of 336 and 720.

Table 14: Long-term Forecast. Mean Squared Error (MSE) results with context length (C) of 512 and forecast horizon (H) of 336, 720. RouteCast consistently outperforms its backbone variants, Chronos (ZS) and Chronos (FT), across all datasets. **Bold** indicates the best (lowest) MSE for each dataset.

Model	ETTh1		ETTh2		ETTm1		ETTm2		Traffic	
	336	720	336	720	336	720	336	720	336	720
Chronos (ZS)	1.224	1.583	0.401	0.486	0.840	1.339	0.366	0.508	0.301	0.346
Chronos (FT)	1.192	1.618	0.429	0.577	0.614	**0.784**	0.378	0.521	0.319	0.362
RouteCast (Ours)	**0.977**	**1.445**	**0.347**	**0.456**	**0.598**	0.830	**0.259**	**0.402**	**0.278**	**0.334**

Experiments are conducted on widely used benchmark datasets, including ETT and Traffic. We exclude the Electricity dataset, as it was part of the pretraining corpus of Chronos and may introduce potential leakage. Overall, this appendix aims to complement our main results by providing a leakage-free evaluation in standard long-term forecasting scenarios while maintaining fair comparison with existing approaches.

Table 14 presents the long-term forecasting results with a context length of 512 and forecast horizons of 336 and 720. Across all benchmark datasets, RouteCast consistently improves over the zero-shot backbone, Chronos (ZS), demonstrating that the proposed framework remains effective even in challenging long-horizon prediction settings. Furthermore, compared to the fine-tuned variant, Chronos (FT), RouteCast achieves substantial gains in most experimental settings. These results indicate that the proposed routing-based framework effectively enhances the forecasting capability of the pretrained backbone beyond standard fine-tuning. Overall, the results suggest that RouteCast generalizes well not only to the short-term forecasting scenarios considered in the main paper, but also to widely adopted long-term forecasting benchmarks.

Q Analysis on Context Routing

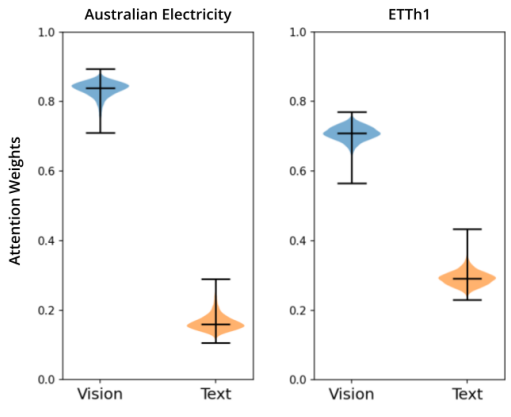


Figure 8: Violin plots of Context routing attention weights for the visual and textual views on the Australian Electricity and ETTh1 datasets on BLIP, Llama configuration.

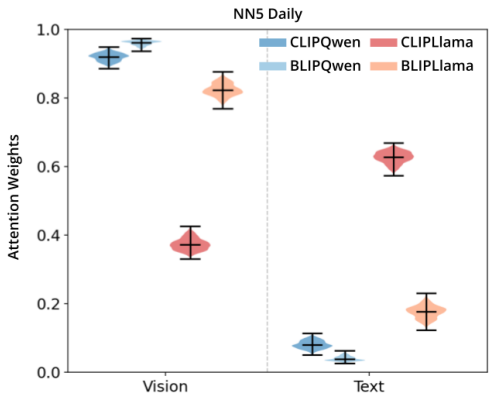
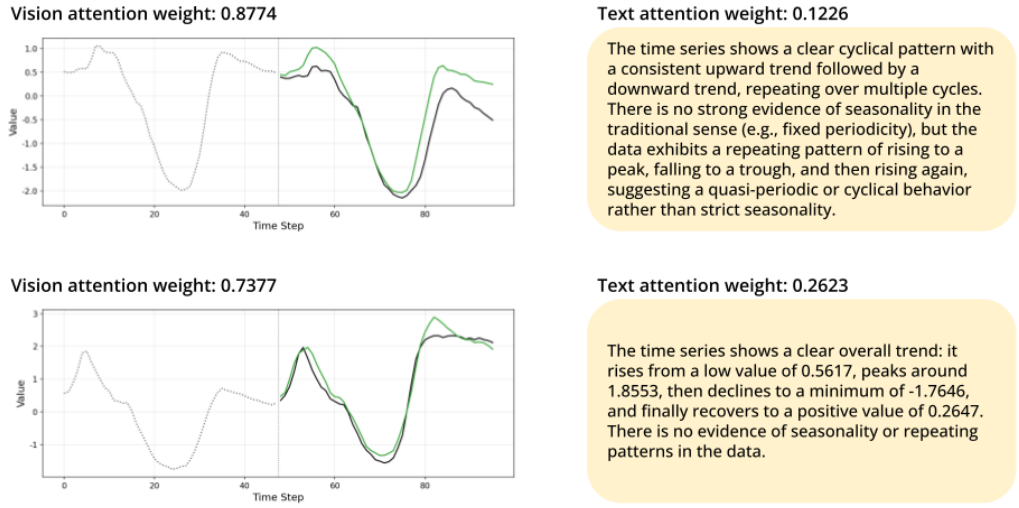


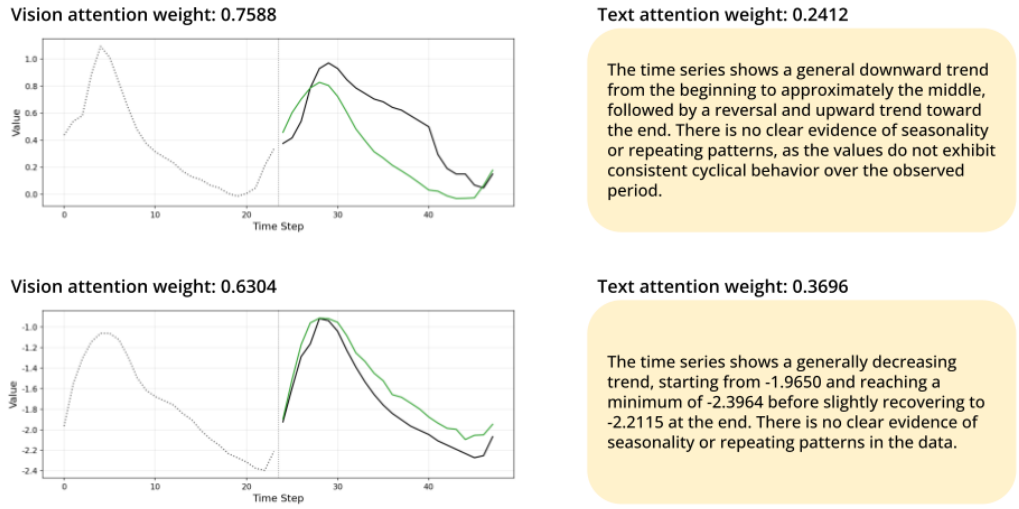
Figure 9: Violin plots of context routing attention weights under different encoder combinations (CLIP, BLIP, Qwen, and Llama).

To verify whether the proposed context routing mechanism operates as intended, we analyze the attention weights produced by the context routing layer. Specifically, we investigate how the routing module allocates attention across views under different datasets, encoder configurations, and forecasting scenarios.

Figure 8 presents violin plots of routing attention weights for the visual and textual views on the Australian Electricity and ETTh1 datasets. We observe that the relative importance assigned to each view differs across datasets. For example, some datasets exhibit larger attention weights for the



(a) Australian Electricity



(b) ETTh2

Figure 10: Forecasting examples of RouteCast disambiguating similar historical time series patterns through adaptive context routing. By selectively attending to visual and textual views, RouteCast captures divergent future dynamics and generates predictions closer to the ground truth.

visual view, while others rely more heavily on textual information. This suggests that the routing module learns dataset-specific dependencies rather than uniformly utilizing all views.

Figure 9 further analyzes routing behaviors under various encoder combinations, including CLIP, BLIP, Qwen, and Llama. Notably, variants using Qwen assign substantially larger attention weights to the visual view while suppressing the textual view. This observation is consistent with the forecasting performance of the Qwen text-only variant reported in Table 4, indicating that the routing module identifies the corresponding text representations as less informative for forecasting. In contrast, variants using Llama exhibit relatively larger attention weights on the textual view, suggesting that the routing module actively utilizes textual information when it provides useful forecasting signals. Overall, these results demonstrate that context routing functions as an adaptive gating mechanism that amplifies informative signals while suppressing noisy or less relevant representations.

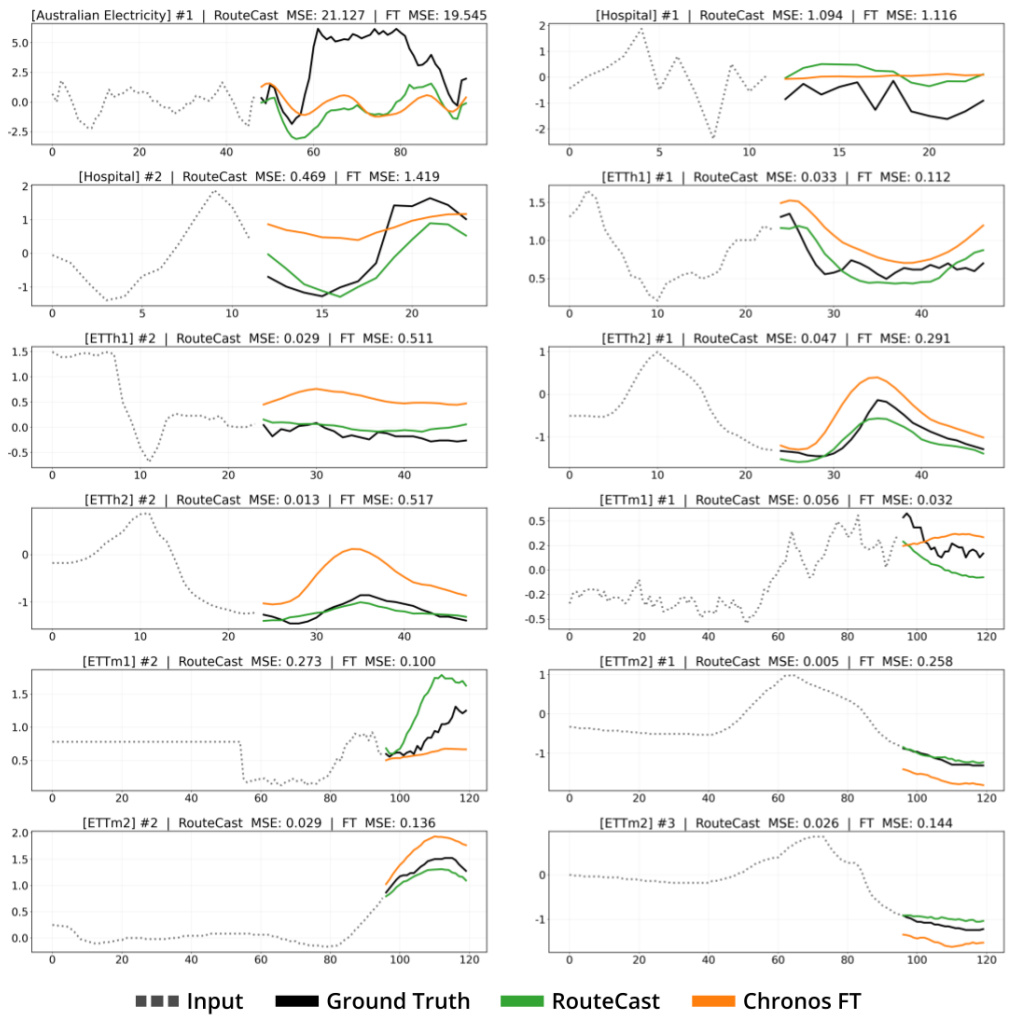
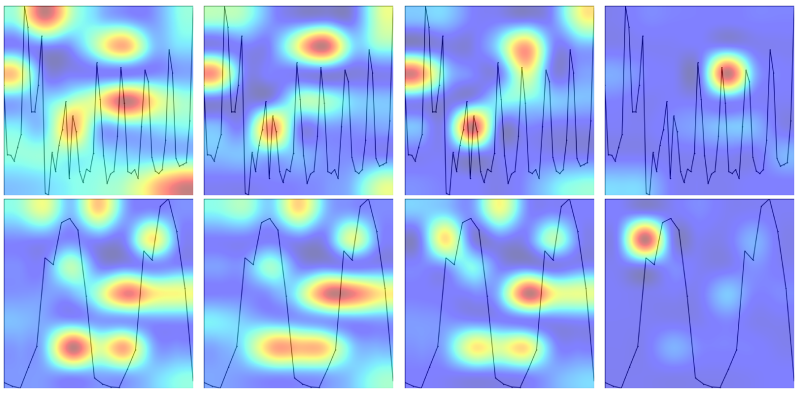


Figure 11: Qualitative Analysis. Qualitative comparison of forecasts on the Australian Electricity, Hospital, and ETT datasets using a Chronos backbone. We contrast RouteCast with fine-tuned Chronos (FT). Across datasets, RouteCast more faithfully captures temporal patterns and aligns them more closely with the ground truth. Notably, even in cases where RouteCast yields a higher MSE than FT, it often produces forecasts that better reflect the underlying structure of the true signal.

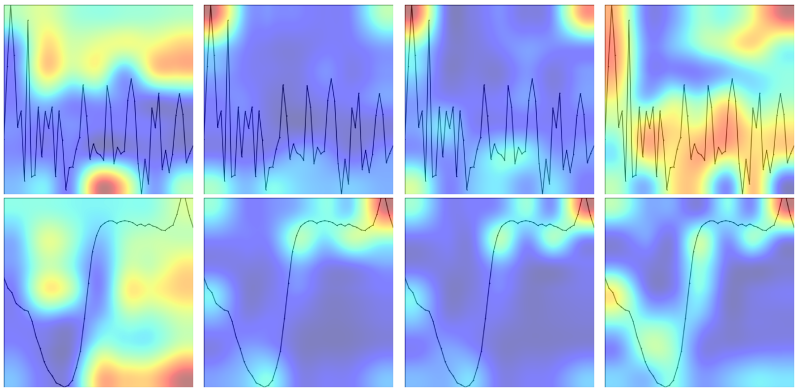
Figure 10 provides qualitative examples where the input time series exhibit highly similar historical patterns despite having substantially different future trajectories. In such cases, relying solely on temporal patterns can lead to ambiguous predictions. RouteCast addresses this challenge by selectively attending to visual and textual views with different routing weights, enabling the model to distinguish between similar historical contexts that evolve into different futures. As a result, RouteCast produces predictions that more closely align with the ground truth trajectories. These observations suggest that context routing serves not merely as a static fusion module, but as a context-aware mechanism for adaptive view selection and temporal disambiguation.

R Qualitative Analysis

We provide a qualitative comparison between RouteCast (built on the Chronos backbone using BLIP and Llama) and a fine-tuned Chronos (FT) model across multiple datasets, including Australian Electricity, Hospital, and ETT (h1, h2, m1, m2). The results are visualized in Figure 11.



(a) BLIP on NN5 Daily and Tourism Monthly



(b) CLIP on NN5 Daily and Australian Electricity

Figure 12: Vision Attention Heatmaps. For each subfigure, the feature map is split into four panels showing the attention at different layers from early to later layers.

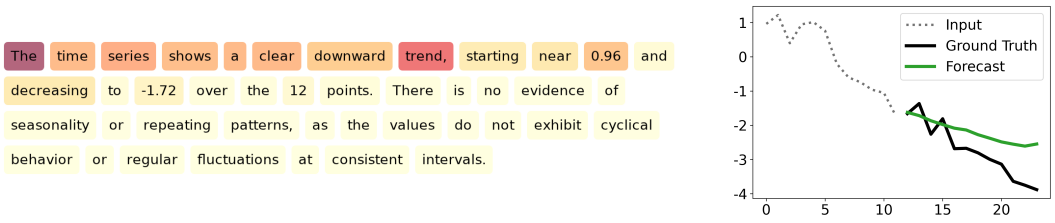
In Hospital Dataset examples, RouteCast demonstrates both lower MSE and a closer match to the ground-truth trajectories compared to FT, accurately capturing salient temporal patterns. In contrast, Australian Electricity and ETTm1 dataset examples highlight cases where RouteCast attains a higher MSE than FT. Nevertheless, its forecasts more faithfully reflect the underlying structure and temporal dynamics of the true signal.

Across the remaining examples, both models often capture similar high-level patterns. However, FT tends to misalign its predictions with respect to the ground-truth scale or offset, whereas RouteCast consistently produces forecasts that are better calibrated and positioned closer to the true values.

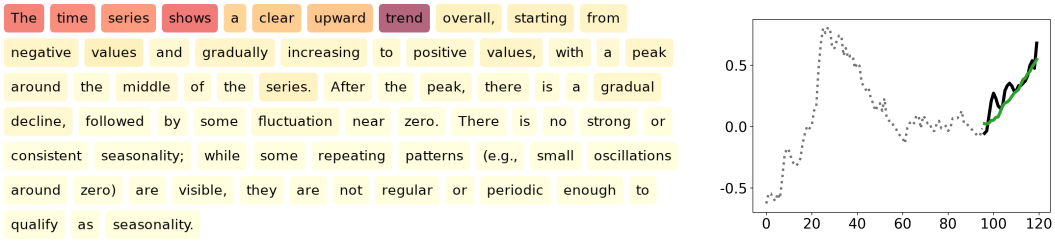
Overall, these observations suggest that RouteCast not only achieves strong quantitative performance but also exhibits a more robust understanding of temporal dynamics, generating predictions that remain structurally coherent with the input time series even in cases where standard error metrics may not fully reflect this advantage.

S HEATMAPS

Vision To qualitatively analyze how RouteCast leverages visual information for time-series forecasting, we visualize layerwise attention heatmaps from the visual view. Figure 12 shows the attention heatmaps of BLIP and CLIP, which are pretrained vision models for RouteCast. For BLIP, we observe a clear progression in attention behavior across layers. In the early layers, the model attends broadly to the overall visual context, capturing global structures and trends. As the layers deepen, attention becomes increasingly concentrated on specific peak regions that are most relevant to the target time-series prediction. This transition from global context modeling to focused, task-relevant regions suggests that RouteCast progressively refines visual information, filtering out



(a) CIF 2016



(b) ETTm1

Figure 13: Text attention heatmaps. For each dataset, the left panel displays the attention heatmap across all layers of the text model for the corresponding textual description, while the right panel shows the forecast results. These visualizations illustrate how the text model attends to different parts of the input and how it contributes to the prediction.

irrelevant details and emphasizing salient cues for forecasting. In contrast, CLIP exhibits a different attention pattern (shown in the appendix). Early layers tend to focus on regions outside the plotted signal, while attention gradually shifts toward the plot area in later layers. Despite these differences, both backbones demonstrate a consistent trend of aligning visual attention with regions relevant to time-series prediction in deeper layers, indicating that RouteCast effectively guides pretrained vision models toward task-specific reasoning.

Text To understand how RouteCast utilizes textual information, we visualize attention heatmaps over text tokens together with the corresponding forecasting outputs. As shown in Figure 13, when the text description contains trend-related phrases such as "upward trend" or "downward trend", the model assigns higher attention weights to these tokens. This attention pattern indicates that RouteCast identifies semantic cues describing temporal dynamics from the textual view. Consistent with this observation, the forecasting outputs exhibit trend behaviors aligned with the textual descriptions, producing predictions that follow upward or downward trajectories accordingly. These results suggest that the textual view contributes high-level temporal guidance, particularly for modeling global trends in time-series forecasting. Overall, the visualization demonstrates that RouteCast effectively links semantic trend information in text descriptions to forecast dynamics.

Time Series To understand how RouteCast utilizes temporal information during forecasting, we visualize attention heatmaps over time-series tokens. Figure 14 reveals that the model does not exclusively focus on the last observed token, which is commonly emphasized in autoregressive forecasting models. Instead, RouteCast selectively attends to multiple temporally relevant tokens, including intermediate positions that appear informative for prediction. This behavior suggests that the model identifies salient temporal patterns and dependencies across the sequence, rather than relying solely on the most recent observation. Such selective attention over time supports more effective temporal context modeling, particularly when important signals occur earlier in the sequence.

T Parameter Efficiency and Computational Cost

In this section, we present the quantitative breakdown of parameter counts and computational resource consumption for RouteCast.

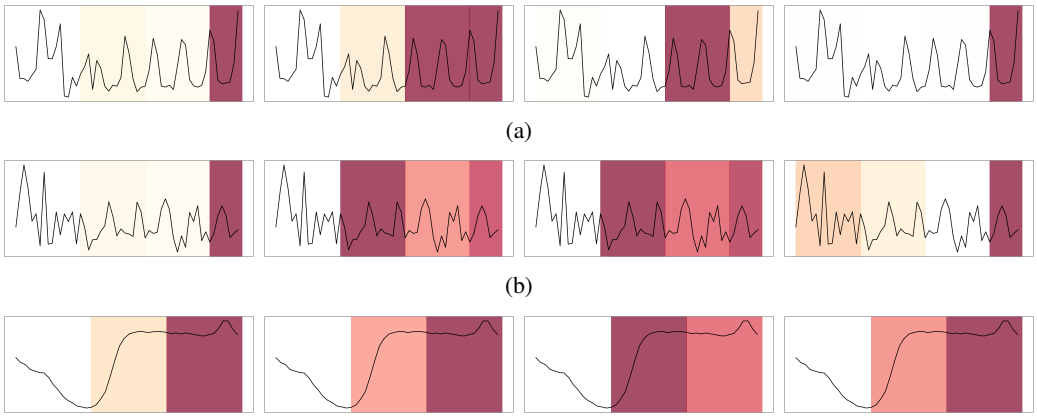


Figure 14: Time-series attention heatmaps. These visualizations show the attention of Chronos to input time-series patches. Subfigures (a) and (b) correspond to the NN5 Daily dataset, while subfigure (c) corresponds to the Australian Electricity dataset. Each subfigure is split into four panels, showing attention at different layers from early to later layers, illustrating how the model progressively refines its focus across the network.

Table 15: **Quantitative comparison of trainable parameter counts and ratios.** The TSFM backbone, vision encoder, and text encoder remain frozen, and only the CP and MR modules are optimized in RouteCast.

Model	VE	TE	# Trainable Params	# Total Params	Trainable Ratio (%)
Chronos			205,292,928	205,292,928	100.00%
RouteCast	CLIP	Qwen	115,356,672	1,951,819,904	5.91%
RouteCast	BLIP	Qwen	115,356,672	1,950,454,400	5.91%
RouteCast	CLIP	LLaMA	395,948,032	7,296,040,576	5.43%
RouteCast	BLIP	LLaMA	395,948,032	7,294,675,072	5.43%

Parameter Efficiency Analysis Table 15 presents the trainable parameter counts, total parameter counts, and trainable ratios across various visual and language encoder configurations compared to the baseline backbone (Chronos). While full fine-tuning Chronos requires optimizing all 205.3M parameters (100%), ROUTECAST restricts updates to 115.4M parameters (5.91%) when paired with Qwen, and 395.9M parameters (5.43%) when paired with LLaMA.

These results demonstrate that ROUTECAST achieves high parameter efficiency during training. By keeping the TSFM backbone, vision encoder, and text encoder entirely frozen and optimizing only the Conditional Prompt (CP) and Modality Routing (MR) modules, the framework substantially reduces training overhead while effectively leveraging strong pretrained representations.

Computational Overhead Trade-off Table 16 summarizes the computational complexity in terms of FLOPs and inference latency. Compared to the standalone Chronos baseline (0.406 GFLOPs, 26.054 ms), ROUTECAST consumes 51.838 GFLOPs and introduces an average inference latency of 82.640 ms (std: 10.492 ms).

Table 16: **Computational cost comparison** between the baseline backbone (Chronos) and RouteCast in terms of FLOPs and inference latency.

Model	GFLOPs	Inference Latency (ms)	
		Mean	Std
Chronos	0.406	26.054	3.176
RouteCast	51.838	82.640	10.492

This indicates that the architectural advantage of ROUTECAST primarily lies in *parameter efficiency* during training rather than computational efficiency during inference. The increased FLOPs and latency reflect the natural computational trade-off required to process multimodal representations from external vision and language encoders in exchange for enhanced predictive accuracy.