CRITICAL-LAYER CONTEXT ALIGNMENT FOR LARGE LANGUAGE MODEL-BASED TIME-SERIES FORECASTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) can forecast time series with their pretrained weights frozen. Existing approaches make numerical inputs compatible with LLMs, but leave open where to supply context and how to train for autoregressive forecasting. Similar local patterns can occur in different window states and at different times. Feeding predicted segments back into the input window also changes the history used for subsequent predictions. To use the overall state of the historical window and the corresponding time information when processing local patches, and to better adapt training to autoregressive forecasting, we propose CLCA (Critical-Layer Context Alignment). A window summary adjusts input patch embeddings, while adapters in selected layers of the fully frozen backbone combine timestamp representations with the current series state. For long-term forecasting, we feed predictions back during training and supervise subsequent segments across different forecast lengths, so that training covers the input conditions encountered during inference. Experiments on eight long-term benchmarks show overall state-of-the-art performance among the compared LLM-based and non-LLM methods under the reported protocols. Ablations show that combining the two context pathways and supervising predictions from windows containing earlier model outputs both improve forecasting performance. With adapter counts matched, adaptation in the final six layers performs best among the tested placements. Short-term, zero-shot, and alternative-backbone experiments further support the applicability of the context-adaptation design.

1 INTRODUCTION

Time-series forecasting depends on both numerical patterns and when they occur. Numerical forecasters use sparse attention, decomposition, and frequency-domain modeling (Zhou et al., 2021; Wu et al., 2021; Zhou et al., 2022b), alongside linear mappings, patching, and cross-variable interactions (Zeng et al., 2023; Nie et al., 2023; Liu et al., 2024c). Pretrained language models provide another approach: numerical tokenization, input reprogramming, and autoregressive prediction allow their sequence representations to be used for forecasting (Zhou et al., 2023; Jin et al., 2024; Liu et al., 2024d). When the language backbone is frozen, an important design question is how to supply the numerical and temporal information needed for the task.

Local observations and window-level context play different roles in forecasting. A similar rising patch can occur within a sustained trend or a fluctuating window, so its local shape alone does not specify the surrounding state of the series. Timestamp representations identify each patch's calendar interval, shared by variables whose numerical histories can differ. A window summary adjusts patch inputs to the surrounding state, while layer-wise adapters use numerical and temporal context according to each patch's hidden state.

We propose Critical-Layer Context Alignment (CLCA), combining window-summary input adaptation with patch-aligned context adapters (Figure 1). CLCA builds on AutoTimes' autoregressive segment prediction with frozen language models (Liu et al., 2024d) and Deep Temporal-Prior Conditioning (Deep TPC), which introduces temporal priors through fusion tokens and gated cross-attention (Bellos et al., 2026). AdapterDrop's adapter-depth analysis (Rücklé et al., 2021) and

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.

MSEF's cross-layer fusion of time-series representations (Chen et al., 2025) motivate our placement study. With adapter counts matched, we identify and validate the final six layers as an effective choice.

Autoregressive forecasting also raises a training issue. AutoTimes trains next-segment prediction using observed inputs, but feeds predicted segments back into the input window during inference. To cover these inference-time inputs, we also feed predictions back during training. CLCA uses one 96-step predictor for different forecast lengths, recomputes context from each updated window, and supervises the subsequent predictions.

Our contributions are:

- We introduce two context pathways: a window summary adjusts patch inputs, and layer-wise adapters incorporate numerical and temporal context. Component ablations support their joint use.
- We identify and validate context adaptation in the final six layers of a complete frozen backbone. The full CLCA framework achieves overall state-of-the-art performance with this placement in our comparisons.
- We train across forecast lengths by feeding predictions back and supervising subsequent segments, covering inference-time inputs. Forecasting benchmarks and targeted ablations validate the method's effectiveness.

2 Related Work

Numerical and pretrained forecasting. TimesNet models intra- and inter-period variation through two-dimensional transformations (Wu et al., 2023). Chronos, TimesFM, and Moirai learn transferable forecasting representations from numerical sequence collections (Ansari et al., 2024; Das et al., 2024; Woo et al., 2024). For language-model adaptation, TEMPO prompts decomposed sequences (Cao et al., 2024), TEST aligns time-series embeddings with text prototypes (Sun et al., 2024), and LLM4TS first aligns time-series representations and then fine-tunes the model for forecasting (Chang et al., 2025).

Context alignment and sparse adaptation. Context can come from domain instructions in UniTime (Liu et al., 2024b) or historical lookback–future examples in ICTSP, which does not require a pretrained LLM (Lu et al., 2025). TimeCMA aligns numerical and prompt representations, CALF constrains cross-modal fine-tuning, and SE-LLM enhances time-series semantics through adaptation (Liu et al., 2025a;b; 2026). Bottleneck adapters, low-rank weight updates, and learnable prompts with zero-initialized attention offer different adaptation mechanisms (Houlsby et al., 2019; Hu et al., 2022; Zhang et al., 2024).

Autoregressive and multi-step training. Autoregressive forecasting feeds predictions back as inputs (Ben Taieb & Hyndman, 2014). AutoTimes learns next-segment prediction with a language backbone (Liu et al., 2024d), whereas Timer-XL and Time-MoE learn autoregressive models from numerical time series (Liu et al., 2025c; Shi et al., 2025). Scheduled Sampling exposes models to generated histories (Bengio et al., 2015); QDF instead learns dependencies and weights among future-step errors (Wang et al., 2026a). Appendix F provides further discussion.

3 Method

Given a history $\mathbf{X} \in \mathbb{R}^{L \times C}$ of L time steps and C variables, we predict the next h time steps:

$$\widehat{\mathbf{Y}}_{1:h} = f_\theta(\mathbf{X}, \mathbf{T}_x, \mathbf{T}_y; h) \in \mathbb{R}^{h \times C}. \tag{1}$$

Here, f_θ is the complete CLCA predictor with trainable parameters θ. The timestamps $\mathbf{T}_x, \mathbf{T}_y$ describe the calendar intervals of the observed and future windows; they contain no future target values. For long-term forecasting, we use the same pretrained language backbone to process each variable's history independently. We use lookback $L = 672$, patch length $P = 96$, and forecast lengths $\mathcal{H} = \{96, 192, 336, 720\}$.

After predicting a segment, we append it to the input window, remove the oldest segment of equal length, and advance the timestamps. The model then continues forecasting from the updated window and its aligned temporal context.

CLCA combines window-summary input adaptation with context adapters in a small subset of frozen language-model layers, then repeatedly predicts future segments (Figure 1). The main backbone is GPT2-774M (Radford et al., 2019). We train the patch encoder and decoder, context modules, fusion tokens, selected-layer adapters, and residual forecasting path. Every token passes through the complete backbone; additional context is injected only at the selected layers.

3.1 WINDOW-LEVEL INPUT ADAPTATION

The patch encoder processes local segments; a shared numerical window summary lets similar local patterns receive different representations when their surrounding window states differ. This summary complements, rather than replaces, the local observations encoded in each patch. For variable c, we normalize the input window using its mean and standard deviation (Kim et al., 2022), then divide it into $N = \lceil L/P \rceil$ non-overlapping patches $\mathbf{p}_{i,c} \in \mathbb{R}^P$, with $P = 96$. With d denoting the backbone's hidden width, a trainable multilayer perceptron $E_{\text{patch}} : \mathbb{R}^P \to \mathbb{R}^d$ encodes each patch, followed by layer normalization and a bounded learnable scale s_e.

Let $\mathbf{r}_c$ collect the window's numerical descriptors, including window statistics and recent changes. Learnable mappings convert these features and metadata into context tokens $C(\mathbf{q}_c)$, which are fused by an MLP and layer normalization to produce the window summary $\mathbf{q}_c \in \mathbb{R}^d$. Thus, $C(\mathbf{q}_c)$ denotes tokens used to construct the summary, not tokens generated from it. Variables sharing calendar intervals can have different summaries. Appendix A.4 specifies the features and mappings; timestamp embeddings enter the layer-wise pathway separately.

A $d \to r \to d$ bottleneck A_s, with bottleneck width r, maps the window summary to a residual that is added to every patch:

$$\begin{aligned} \mathbf{e}_{i,c} &= s_e \, \mathrm{LN}(E_{\text{patch}}(\mathbf{p}_{i,c})), & 0 \le s_e \le s_e^{\max}, \\ \overline{\mathbf{e}}_{i,c} &= \mathbf{e}_{i,c} + \alpha_s A_s(\mathbf{q}_c), & |\alpha_s| \le \alpha_s^{\max}. \end{aligned} \tag{2}$$

Layer normalization has no affine parameters, and the scales s_e and α_s are bounded as shown above; α_s controls the strength of the summary residual. The adjusted patch embeddings combine local patch information with the state of the current input window. When autoregressive forecasting replaces part of the history with generated values, we recompute the normalization statistics and summary from that updated window. The training procedure in Section 3.3 supervises predictions made with these changing inputs.

3.2 LAYER-WISE CONTEXT ADAPTATION

The input summary applies a common adjustment to all patches. Within the backbone, each patch can instead use context according to its own hidden state. We implement this distinction with patch queries that assign position-dependent weights to a shared context sequence. Calendar information identifies when observations occur, while numerical descriptors describe the series at those times. Combining them gives the adapter access to both the timing and the state of the input window.

We use a frozen pretrained language model (GPT2-774M in the main configuration) to encode short texts describing each patch's start and end times offline, and store the hidden state of the last non-padding token as its timestamp embedding. During forecasting, these timestamp vectors are projected when necessary, normalized, and repeated across variables. They advance with the numerical patches as the input window moves, so text encoding is not part of the forecasting loop.

The adapter reads an external context sequence containing timestamp vectors $\boldsymbol{\tau}_c$, the context tokens $C(\mathbf{q}_c)$ defined above, and projections $R(\mathbf{r}_c)$ of the numerical descriptors. All three groups have hidden width d and are concatenated along the token dimension:

$$\mathbf{z}_c = [\, \boldsymbol{\tau}_c \,|||\, C(\mathbf{q}_c) \,|||\, R(\mathbf{r}_c) \,]. \tag{3}$$

The window summary aggregates the tokens in $C(\mathbf{q}_c)$ into a single vector for input adaptation. Keeping the constituent tokens also gives the layer-wise adapters access to their separate representations, allowing each patch to weight individual contextual cues. The sequence $\mathbf{z}_c$ supplies

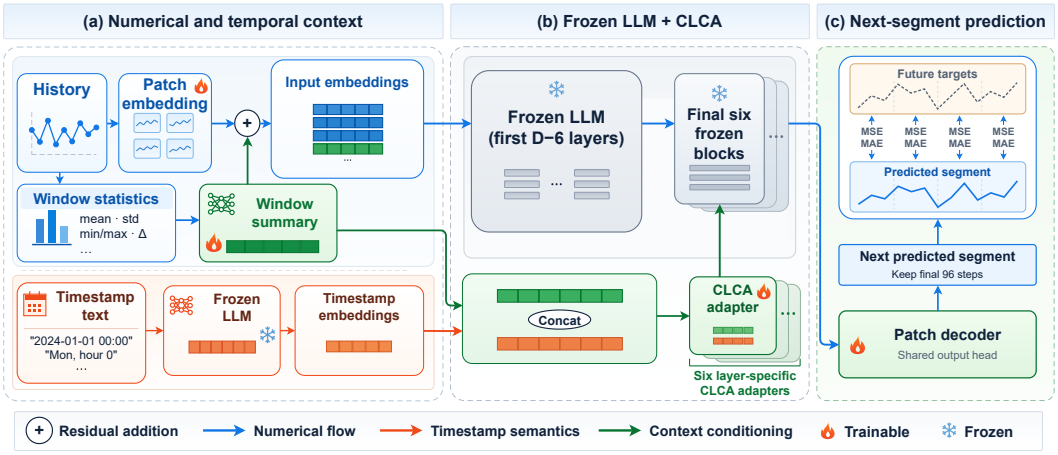

Figure 1: Overview of CLCA. (a) Window summaries adjust patch embeddings, while numerical and timestamp features form the layer-wise context. (b) Adapters introduce context into the final six layers of a fully frozen backbone. (c) A shared output head produces the next 96-step segment; training loss uses only predictions within the requested horizon.

cross-attention keys and values; it is not appended to the backbone input. We append learnable fusion tokens conditioned on the window summary to the end of the patch embedding sequence to aggregate numerical and temporal context.

Two attention paths update these token groups. Following Deep TPC (Bellos et al., 2026), fusion tokens query $\mathbf{z}_c$ through gated cross-attention, using the standard attention operator (Vaswani et al., 2017). A separate patch-state update path queries the same context from individual patch states. The paths have separate query projections: each normalizes its token states, forms queries, and adds a gated residual to the corresponding states. A bounded scale further controls the patch residual. Attention weights can differ across positions, making context directly available to the states used for prediction. Neither path changes $\mathbf{z}_c$.

At each selected layer, the frozen GPT block runs first, followed by the two context-attention paths. The updated patch and fusion-token states retain their original order. An adapter-specific feed-forward network then normalizes and transforms this sequence, adding a sigmoid-gated residual to both groups. This branch is additional to the self-attention and feed-forward computations of the frozen block. Writing the combined adapter residual as A_ℓ, the layer update is

$$\begin{aligned} \widetilde{\mathbf{h}}_c^{\ell} &= \mathrm{GPTBlock}_\ell(\mathbf{h}_c^{\ell}), \\ \mathbf{h}_c^{\ell+1} &= \widetilde{\mathbf{h}}_c^{\ell} + A_\ell(\widetilde{\mathbf{h}}_c^{\ell}, \mathbf{z}_c). \end{aligned} \tag{4}$$

Here, $\mathbf{h}_c^{\ell}$ is the sequence of patch and fusion-token states entering layer ℓ, and $\widetilde{\mathbf{h}}_c^{\ell}$ is its output from the frozen block.

Among patch tokens, each patch attends directly only to itself and preceding patches. All patches can also attend to fusion tokens, which can read the full input sequence. Later blocks can therefore pass fusion updates to patch states. The direct path updates the patch states used for decoding, allowing the final adapter to influence the forecast without requiring a subsequent backbone block to transfer information from fusion tokens. All of this context comes from the current input window.

For a backbone with D blocks, we place the adapters in the final six:

$$\mathcal{S}_{\mathrm{late6}} = \{D-6, D-5, \ldots, D-1\}. \tag{5}$$

Adapters run only at $\ell \in \mathcal{S}_{\mathrm{late6}}$, while every token still traverses all D frozen blocks. Appendix C.1 specifies the fixed layer sets used in the placement comparisons.

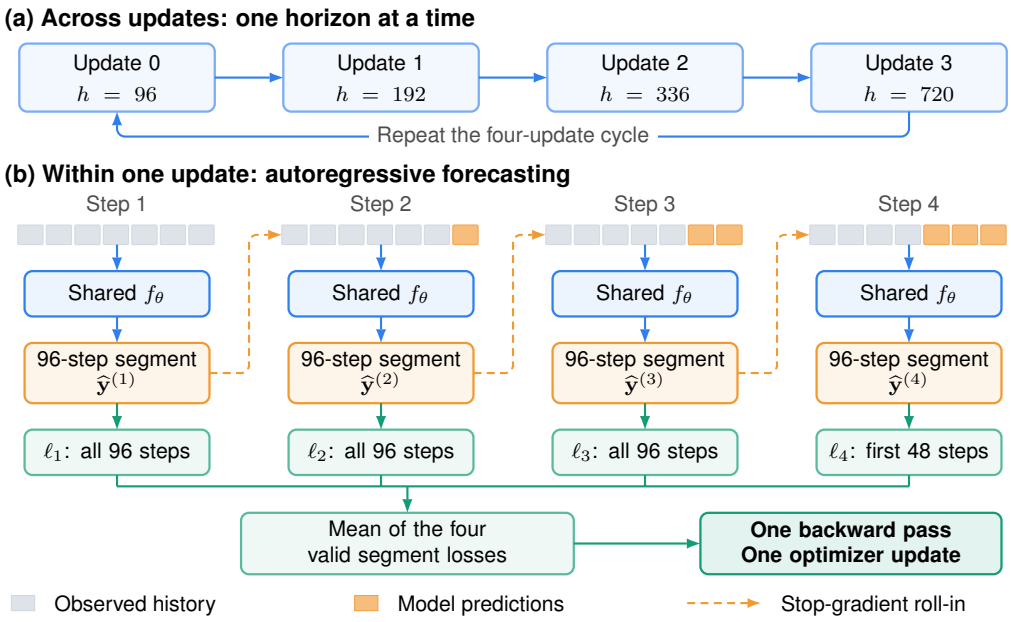

Figure 2: Balanced autoregressive training cycles through four horizons. For 336 steps, the 96-step predictor is called four times; each prediction is detached before reuse as input. Only the first 48 targets in the final segment contribute to the loss. Segment losses are averaged for one update. Sub-batch details are in Appendix A.4.

3.3 Autoregressive Forecasting and Training

CLCA supervises successive predictions across multiple horizons, training the context pathways and output mappings on both the initial observed window and later windows containing model predictions (Figure 2).

The patch decoder applies the same MLP to each final patch state, producing 96 normalized values per patch. We concatenate the patch outputs, add a same-length linear residual of the normalized history, restore the scale, and retain the final 96 values. The decoder and linear residual mapping each reuse their parameters across calls and horizons.

For horizon h, autoregressive forecasting (Ben Taieb & Hyndman, 2014) requires $K_h = \lceil h/P \rceil$ predicted segments. Between calls, the generated segment replaces the oldest part of the history. We refer to this feedback operation as a *roll-in*. The timestamps advance to match the updated window, and its normalization statistics and numerical context are recomputed. Each predicted segment is detached before it is fed back, so later losses do not propagate through generated input values. They still provide gradients to the shared trainable modules in each later call's own forward pass.

Each training update selects one horizon by cycling deterministically through $\mathcal{H}$. To control memory use during multi-step training, we set a sample-count cap for each forecast length and select training samples from the current batch accordingly. Here, balanced training means that horizons are selected equally often. For the selected horizon, each predicted segment is compared with its corresponding future targets. If the final segment extends beyond h, only its valid prefix contributes to the loss. With $\ell(\widehat{\mathbf{y}}, \mathbf{y}) = \mathrm{MSE}(\widehat{\mathbf{y}}, \mathbf{y}) + \lambda_{\mathrm{MAE}} \mathrm{MAE}(\widehat{\mathbf{y}}, \mathbf{y})$, we minimize

$$\mathcal{L}_{\mathrm{roll}}(h) = \frac{1}{K_h} \sum_{k=1}^{K_h} \ell(\widehat{\mathbf{y}}^{(k)}, \mathbf{y}^{(k)}). \tag{6}$$

Here, $\widehat{\mathbf{y}}^{(k)}$ and $\mathbf{y}^{(k)}$ are the valid predictions and targets for segment k, and λ_{MAE} weights the MAE term. Parameters remain unchanged during the successive prediction calls. We average the valid segment losses and then perform one backward pass and one optimizer update. Averaging keeps the aggregate loss scale from increasing simply because a longer horizon requires more calls.

For example, $h = 336$ requires four calls to the 96-step predictor. The first three have 96 valid targets each, and the fourth has 48. Equation 6 gives equal weight to these four segment losses. The input also changes across these calls: the 672-step windows contain 0, 96, 192, and 288 predicted values, respectively, with the remaining positions occupied by observations. At each call, we recompute context from the current window and supervise the resulting prediction. This trains the context pathways and output mappings on histories containing progressively more model predictions, covering the changing input conditions encountered during forecasting.

Algorithms 1 and 2 in Appendix A.4 give the full forward and training procedures. The *96-only training* comparison in Section 4.3 evaluates the contribution of supervising predictions beyond the first segment.

4 EXPERIMENTS

Datasets. We evaluate CLCA on eight long-term forecasting datasets: ETTh1, ETTh2, ETTm1, ETTm2, ECL, Weather, Traffic, and Solar. We also evaluate short-term forecasting on M4 and zero-shot forecasting in both directions between M4 and M3 (Makridakis et al., 2020; Makridakis & Hibon, 2000). Appendix A.3 gives dataset details and splits.

Experimental setup. Long-term experiments use a 672-step lookback and one shared 96-step predictor. We select checkpoints based on validation loss in all our experiments. At each test origin, we generate one 720-step autoregressive forecast and compute errors over its first 96, 192, 336, and 720 steps (shared-prefix evaluation). We report mean squared error (MSE) and mean absolute error (MAE) for long-term forecasting; symmetric mean absolute percentage error (sMAPE), mean absolute scaled error (MASE), and overall weighted average (OWA) for M4; and sMAPE for M4–M3 zero-shot forecasting. Appendix A.2 defines the metrics, normalization, cropping, and evaluation.

Results from our experiments in Table 1 and all variants in Tables 4 and 5 are averaged over three random seeds. The corresponding appendix tables include sample standard deviations; Appendix A.1 gives statistical and reporting details. Best and second-best results are bolded and underlined, respectively; component ablation tables mark only the best.

4.1 LONG-TERM FORECASTING

We compare CLCA with LLM-based methods—AutoTimes (Liu et al., 2024d), Time-LLM (Jin et al., 2024), CALF (Liu et al., 2025b), UniTime (Liu et al., 2024b), and FPT (Zhou et al., 2023)—and numerical forecasting methods—iTransformer (Liu et al., 2024c), DLinear (Zeng et al., 2023), PatchTST (Nie et al., 2023), and TimesNet (Wu et al., 2023). Most published baseline results come from the AutoTimes one-for-all benchmark; we reproduce the remaining long-term baselines from official code using the same lookback and one-for-all prediction setup (Appendix A.5).

CLCA achieves overall state-of-the-art performance among the LLM-based and numerical methods in Table 1, considering MSE and MAE jointly across datasets. On ETTm1, ECL, and Weather, CLCA achieves lower MSE and MAE than CALF across all four forecast horizons (Appendix B.1). These gains persist from 96 to 720 steps, while the comparison on Traffic primarily favors CLCA in average MAE.

4.2 SHORT-TERM AND ZERO-SHOT FORECASTING

Short-term forecasting. The M4 comparison includes Koopa (Liu et al., 2023) and FiLM (Zhou et al., 2022a), as well as N-BEATS with basis expansion and N-HiTS with hierarchical interpolation (Oreshkin et al., 2020; Challu et al., 2023). CLCA achieves the best overall M4 results among the compared methods in Table 2.

Zero-shot forecasting. The zero-shot comparison includes NSFormer (Liu et al., 2022), FEDformer (Zhou et al., 2022b), Informer (Zhou et al., 2021), and Reformer (Kitaev et al., 2020). CLCA achieves the lowest reported sMAPE in both M4–M3 directions in Table 3. Models are trained on the source domain without gradient updates on target-domain samples; Appendix B.3 describes the complete protocol.

Table 1: Long-term forecasting on eight datasets, including Electricity (ECL). Baselines combine AutoTimes reports (Liu et al., 2024d) and official-code reproductions. See Appendix A.5 for protocols.

Method	CLCA		AutoTimes		Time-LLM		CALF		UniTime		FPT		iTransformer		DLinear		PatchTST		TimesNet	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.398	0.423	**0.389**	**0.422**	0.412	0.437	0.393	0.425	0.683	0.596	0.429	0.439	0.421	0.445	0.426	0.444	0.409	0.430	0.495	0.491
ETTh2	**0.317**	**0.376**	0.359	0.403	0.377	0.418	0.324	**0.376**	0.371	0.409	0.362	0.404	0.390	0.418	0.471	0.469	0.342	0.389	0.430	0.454
ETTm1	**0.351**	**0.377**	0.362	0.391	0.369	0.398	0.365	0.394	0.352	0.387	0.367	0.392	0.375	0.406	0.360	0.384	0.353	0.383	0.470	0.439
ETTm2	0.254	**0.313**	0.275	0.329	0.267	0.326	0.267	0.323	0.261	0.320	0.276	0.332	0.275	0.335	0.266	0.331	**0.250**	**0.313**	0.288	0.343
ECL	**0.153**	**0.248**	0.159	0.253	0.181	0.288	0.160	0.259	0.325	0.399	0.184	0.284	0.164	0.258	0.165	0.265	0.169	0.268	0.201	0.303
Weather	0.226	**0.260**	0.235	0.273	**0.225**	0.266	0.236	0.274	0.461	0.459	0.228	0.266	0.266	0.291	0.239	0.291	0.226	0.268	0.264	0.293
Traffic	0.382	**0.257**	**0.374**	0.264	0.410	0.303	0.382	0.259	0.584	0.367	0.461	0.326	0.384	0.274	0.423	0.298	0.391	0.275	0.602	0.322
Solar	**0.196**	**0.235**	0.197	0.242	0.263	0.335	0.201	0.258	0.392	0.462	0.236	0.303	0.213	0.291	0.222	0.283	0.202	0.269	0.213	0.295

Table 2: M4 short-term forecasting, measured by official overall sMAPE, MASE, and OWA. Baseline results follow AutoTimes.

	Method	CLCA	AutoTimes	Time-LLM	FPT	Koopa	N-HiTS	DLinear	PatchTST	TimesNet	FiLM	N-BEATS
Average	sMAPE	**11.752**	11.831	11.983	11.991	11.863	11.960	12.418	13.022	11.930	12.489	11.910
	MASE	**1.550**	1.585	1.595	1.600	1.595	1.606	1.656	1.814	1.597	1.690	1.613
	OWA	**0.839**	0.850	0.859	0.861	0.858	0.861	0.891	0.954	0.867	0.902	0.862

Table 3: M4–M3 zero-shot forecasting. Average sMAPE is reported for source-only training in each direction (Appendix B.3).

Method	CLCA	AutoTimes	FPT	DLinear	PatchTST	TimesNet	NSFormer	FEDformer	Informer	Reformer
M4→M3	**12.621**	12.750	13.060	14.030	13.390	14.170	15.290	13.530	15.820	13.370
M3→M4	**13.029**	13.036	13.125	15.337	13.228	14.553	14.327	15.047	19.047	14.092

4.3 Ablation studies

Adapter placement. We compare five ways of placing six adapters within the same frozen backbone: the first six layers, the middle six, evenly spaced layers, randomly selected layers, and the final six. Holding the adapter count constant lets us evaluate where context is most effectively introduced. Figure 3 reports Average MSE on ETTh1 and ETTm1.

The final-six-layer placement has the lowest MSE averaged across horizons and seeds, and the smallest observed between-seed spread, on both datasets. With the adapter count held constant, this comparison identifies placement as an important design choice and supports concentrating context adaptation in the later layers. CLCA uses this empirically supported placement within the complete frozen backbone. The all-layer reference also has higher average MSE and MAE than late6 on ETTh1, showing that additional injection sites do not necessarily improve forecasting. Appendix C.1 provides the layer indices, per-horizon results, and the all-layer reference.

Context interfaces and multi-step supervision. The full model attains the lowest Average MSE on ETTh1 and ETTh2 in Table 4. On ETTh2, Average MAE is very similar for Full and the three component-removal variants. The three component removals target the two context pathways: *w/o Context* removes input-summary adaptation, *w/o CLCA* removes the layer-wise branch and its timestamp input, and *w/o Both* removes both. The full model's aggregate MSE advantage supports combining the common window-level input adjustment with position-dependent context adaptation inside the backbone.

The *96-only training* variant retains the full model, per-horizon sample-count caps, and sampling procedure, but each update predicts and supervises only the first 96 steps. The full model's lower aggregate errors support supervising later predictions from windows containing generated values, with context recomputed from each updated window.

Figure 4 plots the relative changes in test error from component removal at each horizon. On ETTh1, all removals increase both metrics, most strongly at 96 steps. On ETTh2, component removal in-

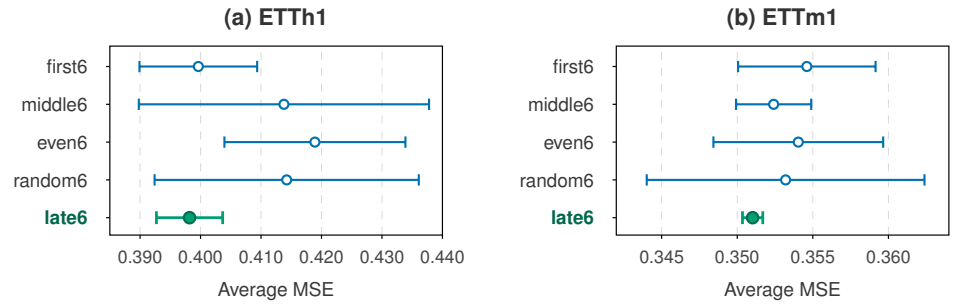

Figure 3: Effect of adapter placement on ETTh1 and ETTm1. Points and error bars show the mean and sample standard deviation of Average MSE across three seeds (Appendix A.1). Each strategy uses six adapters; layer indices and detailed results are in Appendix C.1.

Table 4: Component and training ablations on ETTh1 and ETTh2.

	ETTh1		ETTh2	
Variant	MSE	MAE	MSE	MAE
Full	**0.398**	**0.423**	**0.317**	0.376
96-only training	0.406	0.426	0.322	0.377
w/o CLCA	0.475	0.471	0.323	0.376
w/o Context	0.459	0.463	0.322	**0.375**
w/o Both	0.439	0.449	0.321	**0.375**

creases MSE at 96, 192, and 336 steps; MAE increases at 96 steps but decreases at 336 and 720 steps, partially offsetting across horizons and leaving similar average values. Appendix C provides full results and forecast examples.

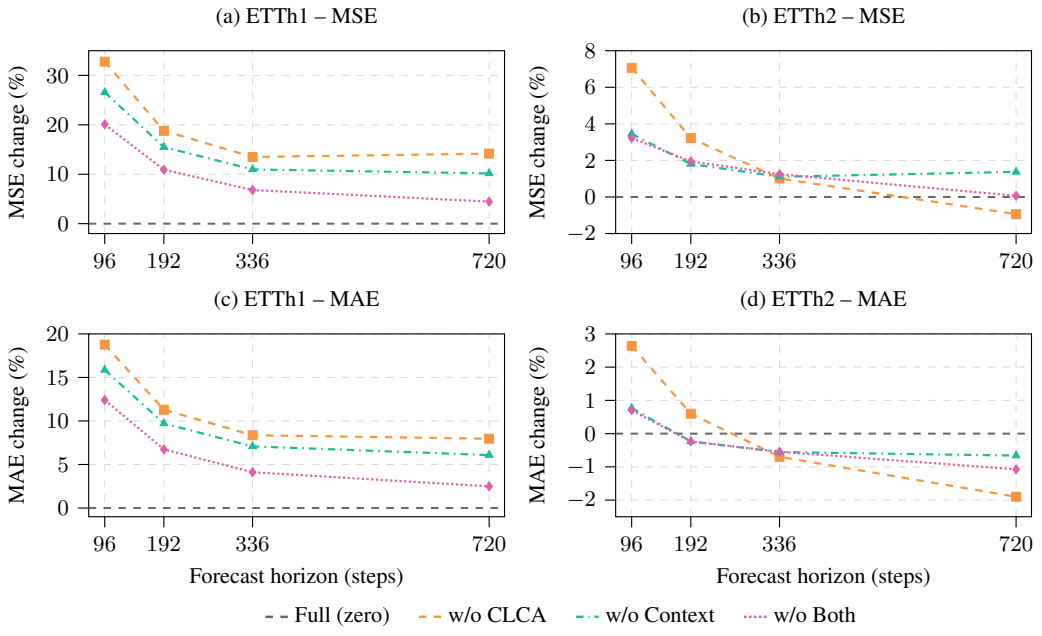

Figure 4: Component ablations on ETTh1 and ETTh2: percentage changes in MSE and MAE relative to Full.

Lightweight backbone replacements. Table 5 compares three replacements for the pretrained language-model backbone (Tan et al., 2024): *w/o LLM* (identity mapping), *LLM2Attn* (one causal multi-head self-attention layer), and *LLM2Trsf* (one causal Transformer layer). The latter two are randomly initialized and trainable. All retain the input summary, timestamp embeddings, and one pre-decoder context adapter; Appendix C.3 gives its cross-attention/TPC computation and optimization settings.

CLCA achieves the best or tied-best Average MSE and MAE on both datasets. The strongest lightweight alternative changes from *w/o LLM* on ETTh1 to *LLM2Attn* on ECL. CLCA's consistently strong results support combining the pretrained backbone with context adaptation across datasets.

Table 5: Lightweight backbone replacements on ETTh1 and ECL. *w/o LLM*, *LLM2Attn*, and *LLM2Trsf* keep the summary branch and one context adapter (Appendix C.3).

Dataset	ETTh1								ECL							
Variant	CLCA		w/o LLM		LLM2Attn		LLM2Trsf		CLCA		w/o LLM		LLM2Attn		LLM2Trsf	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Pred-96	**0.360**	**0.394**	0.366	0.398	0.405	0.428	0.396	0.421	**0.119**	**0.219**	0.127	0.231	**0.119**	0.220	0.120	0.223
Pred-192	**0.383**	**0.410**	0.385	**0.410**	0.405	0.428	0.400	0.424	**0.137**	**0.235**	0.151	0.255	0.138	0.237	0.142	0.242
Pred-336	0.401	0.424	**0.397**	**0.421**	0.410	0.435	0.409	0.433	**0.157**	**0.253**	0.183	0.283	**0.157**	0.255	0.164	0.264
Pred-720	0.449	0.464	**0.448**	**0.462**	0.459	0.473	0.459	0.472	0.201	**0.287**	0.258	0.339	**0.200**	0.290	0.216	0.305
Average	**0.398**	**0.423**	0.399	**0.423**	0.420	0.441	0.416	0.438	**0.153**	**0.248**	0.180	0.277	**0.153**	0.251	0.160	0.258

4.4 Alternative language backbones

We evaluate alternative pretrained backbones to assess whether CLCA's context-adaptation design remains effective across language-model families and model sizes. Table 6 compares CLCA using two GPT2 sizes (Radford et al., 2019), OPT-1.3B and OPT-2.7B (Zhang et al., 2022), and Llama2-7B (Touvron et al., 2023), all using the shared prediction interface.

GPT2-774M performs best on ETTh1, while OPT-2.7B performs best on Weather. Forecasting quality does not simply increase with parameter count; dataset suitability also matters. CLCA achieves strong forecasting performance across GPT2, OPT, and Llama backbones, demonstrating the effectiveness and cross-backbone generality of our context-adaptation approach.

Table 6: Alternative language backbones on ETTh1 and Weather. Configurations are in Appendix D.1.

Backbone	GPT2-774M		GPT2-124M		OPT-1.3B		OPT-2.7B		Llama2-7B	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.398**	**0.423**	0.408	0.429	0.427	0.442	0.417	0.434	0.407	0.430
Weather	0.226	0.260	0.226	0.258	0.227	0.258	**0.223**	**0.257**	0.238	0.267

5 Conclusion

We presented CLCA, which supplies context at the input and within selected layers of a fully frozen language backbone. Long-term benchmarks show strong performance across forecast lengths. Ablations support combining the context pathways and supervising predictions after model outputs are fed back; matched-count comparisons favor the final six layers. These findings guide context placement and training for autoregressive forecasting. Recomputing context at each call lets the same frozen backbone use the evolving series state and time information as predictions enter the input window. Alternative-backbone, short-term, and zero-shot experiments further support the context-adaptation design's applicability. Future work could jointly optimize adapter counts and locations across pretrained language models, incorporate richer contextual inputs, and extend CLCA to provide calibrated uncertainty estimates (Appendix E).

Reproducibility Statement

Implementation details and evaluation protocols are provided in Appendix A, with detailed experimental results in Appendices B–D. The code will be publicly released upon acceptance.

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.

Filippos Bellos, NaveenJohn Premkumar, Yannis Avrithis, Nam H. Nguyen, and Jason J. Corso. Deep TPC: Temporal-prior conditioning for time series forecasting. arXiv preprint arXiv:2602.16188, 2026. Accepted at the 2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP).

Souhaib Ben Taieb and Rob J. Hyndman. Boosting multi-step autoregressive forecasts. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pp. 109–117. PMLR, 2014.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 28, pp. 1171–1179. Curran Associates, Inc., 2015.

Defu Cao, Furong Jia, Sercan O. Arik, Tomas Pfister, Yixiang Zheng, Wen Ye, and Yan Liu. TEMPO: Prompt-based generative pre-trained transformer for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

Cristian Challu, Kin G. Olivares, Boris N. Oreshkin, Federico Garza Ramirez, Max Mergenthaler Canseco, and Artur Dubrawski. NHITS: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 6989–6997. AAAI Press, 2023.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. LLM4TS: Aligning pre-trained LLMs as data-efficient time-series forecasters. *ACM Transactions on Intelligent Systems and Technology*, 16(3):60:1–60:20, 2025.

Zhuomin Chen, Dan Li, Jiahui Zhou, Shunyu Wu, Haozheng Ye, Jian Lou, and See-Kiong Ng. Integrating time series into LLMs via multi-layer steerable embedding fusion for enhanced forecasting. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management*, pp. 4675–4679. ACM, 2025.

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*, volume 235 of *Proceedings of Machine Learning Research*, pp. 10148–10167. PMLR, 2024.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2790–2799. PMLR, 2019.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations*, 2022.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y. 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 *The Twelfth International Conference on Learning Representations*, 2024.

Taesung Kim, Jinhee Kim, Yunwon Tae, Cheonbok Park, Jang-Ho Choi, and Jaegul Choo. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *The Tenth International Conference on Learning Representations*, 2022.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. In *The Eighth International Conference on Learning Representations*, 2020.

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 AAAI Conference on Artificial Intelligence*, volume 39, pp. 18780–18788. AAAI Press, 2025a.

Hao Liu, Xiaoxing Zhang, Chun Yang, and Xiaobin Zhu. Semantic-enhanced time-series forecasting via large language models. In *The Fourteenth International Conference on Learning Representations*, 2026.

Peiyuan Liu, Hang Guo, Tao Dai, Naiqi Li, Jigang Bao, Xudong Ren, Yong Jiang, and Shu-Tao Xia. CALF: Aligning LLMs for time series forecasting via cross-modal fine-tuning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 18915–18923. AAAI Press, 2025b.

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

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. Non-stationary transformers: Exploring the stationarity in time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pp. 9881–9893. Curran Associates, Inc., 2022.

Yong Liu, Chenyu Li, Jianmin Wang, and Mingsheng Long. Koopa: Learning non-stationary time series dynamics with Koopman predictors. In *Advances in Neural Information Processing Systems*, volume 36, pp. 12271–12290. Curran Associates, Inc., 2023.

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. iTransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024c.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. AutoTimes: Autoregressive time series forecasters via large language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 122154–122184. Curran Associates, Inc., 2024d.

Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer-XL: Long-context transformers for unified time series forecasting. In *The Thirteenth International Conference on Learning Representations*, 2025c.

Jiecheng Lu, Yan Sun, and Shihao Yang. In-context time series predictor. In *The Thirteenth International Conference on Learning Representations*, 2025.

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

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

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.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *The Eighth International Conference on Learning Representations*, 2020.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Andreas Rücklé, Gregor Geigle, Max Glockner, Tilman Beck, Jonas Pfeiffer, Nils Reimers, and Iryna Gurevych. AdapterDrop: On the efficiency of adapters in transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7930–7946. Association for Computational Linguistics, 2021.

Xiaoming Shi, Shiyu Wang, Yuqi Nie, Dianqi Li, Zhou Ye, Qingsong Wen, and Ming Jin. Time-MoE: Billion-scale time series foundation models with mixture of experts. In *The Thirteenth International Conference on Learning Representations*, 2025.

Chenxi Sun, Hongyan Li, Yaliang Li, and Shenda Hong. TEST: Text prototype aligned embedding to activate LLM's ability for time series. In *The Twelfth International Conference on Learning Representations*, 2024.

Mingtian Tan, Mike A. Merrill, Vinayak Gupta, Tim Althoff, and Thomas Hartvigsen. Are language models actually useful for time series forecasting? In *Advances in Neural Information Processing Systems*, volume 37, pp. 60162–60191. Curran Associates, Inc., 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. arXiv preprint arXiv:2307.09288, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pp. 5998–6008. Curran Associates, Inc., 2017.

Hao Wang, Licheng Pan, Yuan Lu, Zhichao Chen, Tianqiao Liu, Shuting He, Zhixuan Chu, Qingsong Wen, Haoxuan Li, and Zhouchen Lin. Quadratic direct forecast for training multi-step time-series forecast models. In *The Fourteenth International Conference on Learning Representations*, 2026a.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 53140–53164. PMLR, 2024.

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021.

Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. TimesNet: Temporal 2D-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128. AAAI Press, 2023.

Renrui Zhang, Jiaming Han, Chris Liu, Aojun Zhou, Pan Lu, Yu Qiao, Hongsheng Li, and Peng Gao. LLaMA-Adapter: Efficient fine-tuning of large language models with zero-initialized attention. In *The Twelfth International Conference on Learning Representations*, 2024.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. OPT: Open pre-trained transformer language models. arXiv preprint arXiv:2205.01068, 2022.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 11106–11115. AAAI Press, 2021.

Tian Zhou, Ziqing Ma, Xue Wang, Qingsong Wen, Liang Sun, Tao Yao, Wotao Yin, and Rong Jin. FiLM: Frequency improved Legendre memory model for long-term time series forecasting. In *Advances in Neural Information Processing Systems*, volume 35, pp. 12677–12690. Curran Associates, Inc., 2022a.

Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. FEDformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 27268–27286. PMLR, 2022b.

Tian Zhou, Peisong Niu, Xue Wang, Liang Sun, and Rong Jin. One fits all: Power general time series analysis by pretrained LM. In *Advances in Neural Information Processing Systems*, volume 36, pp. 43322–43355. Curran Associates, Inc., 2023.