Enhancing Zero-Shot Time Series Forecasting in Off-the-Shelf LLMs via Noise Injection Prompting

Xingyou Yin[‡], **Ceyao Zhang**[§], **Min Hu**[‡], **Kai Chen**[‡,*]
[‡]School of Mathematics and Statistics, Central South University,
[§]School of Intelligence Science and Technology, Peking University
{yinxingyou, minhu, kaichen6}@csu.edu.cn, ceyaozhang@pku.edu.cn

Abstract

Large Language Models (LLMs) have demonstrated effectiveness as zero-shot time series (TS) forecasters. While existing work often relies on fine-tuning specialized modules to bridge this gap, a distinct, yet challenging, paradigm aims to leverage truly off-the-shelf LLMs without any fine-tuning whatsoever, relying solely on strategic tokenization of numerical sequences. However, the parameters of these fully frozen models cannot adapt to distribution shifts. Thus, we introduce a novel yet highly effective strategy to overcome this brittleness: injecting noise into the raw TS before tokenization. This non-invasive intervention acts as a form of inference-time augmentation, compelling the frozen LLM to extrapolate based on robust underlying temporal patterns rather than superficial numerical artifacts. We theoretically analyze this phenomenon and empirically validate its effectiveness across diverse benchmarks. Notably, to fully eliminate potential biases from data contamination during LLM pre-training, we introduce multiple novel real-world TS datasets that fall outside all utilized LLMs' pre-training scopes, and consistently observe improved performance. This study provides a further step in directly leveraging off-the-shelf LLMs for TS forecasting[1].

1 Introduction

Time series (TS) modeling plays a critical role in various real-world applications, including climate, economics, energy, and operations (Wu et al., 2021; Liu et al., 2023). Accurate TS forecasting relies on the ability to model complex temporal dependencies in data, such as trends, seasonality, and nonlinearity, and to predict future values based on historical observations (Montgomery et al., 2015; Che et al., 2018). Traditional TS forecasters, such as ARIMA (Box and Jenkins, 1968), nonlinear models (Zhang et al., 2021, 2023), and Gaussian processes (GP; Xu et al., 2020; Dai et al., 2020; Chen et al., 2022), rely on human prior knowledge to select appropriate model configurations, for instance, kernel choices in GPs, to capture the underlying patterns and achieve accurate predictions. While **deep learning** (DL; LeCun et al., 2015) has made impressive advances in NLP and CV, demonstrating that learned features can outperform human-designed features, these DL-based methods (Zeng et al., 2023) have also been extended to the TS domain. However, both traditional methods and DL-based TS methods require training from scratch for a specific TS task. Recently, **large language models (LLMs)**, such as GPT-3 (Brown et al., 2020), have demonstrated the ability to perform downstream tasks without the need for fine-tuning, enabling zero-shot learning. Based on this capability, Gruver et al. (2023) proved that LLMs can serve as zero-shot TS forecasters by tokenizing TS data.

As the use of LLM-based methods for TS forecasting, hereafter referred to as *LLMs-for-TS*, has increased, more strategies have been developed to improve their performance in TS modeling. There are two primary paradigms for TS forecasting of LLMs (Sun et al., 2024). The first paradigm focuses on designing and training a dedicated TS-specific large model from scratch or by fine-tuning an existing pre-trained LLM to transfer it from textual to temporal domains. Representatives include LLM4TS (Chang et al., 2025), which adapts LLMs to TS through autoregressive supervised fine-tuning, and CALF (Liu et al., 2025), which employs a dual-branch architecture with attention-based embeddings. The second paradigm involves freezing the parameters of the existing LLM and designing TS representations that are compatible with them. LLMTime (Gruver et al., 2023; Liu et al., 2024) directly inputs raw numeric TS data as text prompts, and TEST (Sun et al., 2024) aligns temporal instances with textual prototypes. This paradigm fully maximizes the model's capabili-

[1]https://github.com/jkumh/NLTS

Findings of the Association for Computational Linguistics: ACL 2026, pages 41281–41308

ties by aligning the raw input data with the LLM's native architecture, allowing it to process TS effectively without further retraining.

Meanwhile, the term ***zero-shot*** is ambiguous in different LLM-based TS methods; we disambiguate it as follows. The first type is **zero-shot under trainable settings**, which covers all first LLM-for-TS paradigms and parts of the second paradigm, such as TEST, which freeze the LLM but train the other modules. Under this setting, *zero-shot* refers to the model being first fine-tuned on source TS A before inference on target TS B. In comparison, **zero-shot under training-free setting** directly applies LLMs to input queries without any task-specific adaptation or fine-tuning, such as PromptCast (Xue and Salim, 2023), LLMTime (Gruver et al., 2023). We focus on this more challenging yet practically valuable **Off-the-Shelf LLMs** setting, as it most purely and directly tests an LLM's inherent reasoning capabilities and offers the lowest barrier to deployment by eliminating any need for training data or fine-tuning computation.

However, the performance of off-the-shelf LLMs is acutely dependent on the textual representation of the continuous TS data. A central challenge in this paradigm is the brittleness of the tokenized input: the forecasting accuracy can be highly sensitive to the specific numerical representation and vulnerable to distribution shifts, as the model's parameters are frozen and cannot adapt. To address this, we turn to a powerful yet under-explored strategy for enhancing model robustness: input-level **noise injection**. Traditionally employed during model training for regularization and data augmentation (Trirat et al., 2024; Ma et al., 2024; Abnar et al., 2021; Jin et al., 2023b), we reconceptualize its application for inference-time robustness in frozen LLMs. We hypothesize that strategically perturbing the TS with noise before tokenization can act as a powerful form of input augmentation, compelling the LLM to base its predictions on more stable, underlying temporal structures rather than on the precise, and potentially misleading, numerical representation. This approach is fundamentally non-invasive, aligning perfectly with the off-the-shelf paradigm by enhancing performance without any manipulation of model internal parameters or the need for additional retraining efforts.

Translating this hypothesis into a practical framework, we introduce Noise-injected LLM for Time Series (NLTS). Our method operates by first injecting controlled stochastic noise into the raw TS. This perturbed series is then converted into a discrete token sequence through a meticulous textualization and tokenization process that preserves bijectivity between numerical and symbolic representations. This noised, tokenized prompt is fed directly into the frozen LLM, which then performs the autoregressive *next-token prediction*. To aggregate predictions and estimate uncertainty, we sample multiple forecasts by generating varied noisy instances of the input prompt, then compute the median and variance across these samples. This entire pipeline requires no backpropagation, fine-tuning, or internal access to the LLM. Specifically, our contributions are as follows.

- We propose a noise-injection strategy that equips off-the-shelf LLMs with enhanced zero-shot forecasting capabilities without any computational task-specific fine-tuning.
- We provide both theoretical guarantees and exhaustive empirical validation on established benchmarks and settings, demonstrating consistent gains across different LLMs.
- We expose data contamination risks inherent in LLM-based zero-shot forecasting, and subsequently design four new datasets to eliminate this risk. Experiments further validate the effectiveness of our method.

2 Related Work

LLMs for time series forecasting. PromptCast (Xue and Salim, 2023) was the first study to apply pre-trained LLMs to TS forecasting. Subsequently, LLMTime (Gruver et al., 2023) innovatively inputs numeric TS directly into LLMs, transforming TS forecasting into a "next-word prediction" task. This shift has enabled LLMs like GPT-3.5 (Brown et al., 2020)and LLaMA-2 (Touvron et al., 2023) to excel in zero-shot forecasting tasks, further demonstrating the potential of LLMs in TS tasks. On the other hand, significant progress has been made in block-based representation methods for TS (Nie et al., 2023a). Examples include One Fits All (OFA) (Zhou et al., 2023), LLM4TS (Chang et al., 2025), TEST (Sun et al., 2024), TEMPO (Cao et al.), and TimeLLM (Jin et al., 2023a), all of which employ block-based methods to tokenize TS, making them compatible with LLM and improving performance. All these methods overlook the impact of noise injection in the context of LLMs-for-TS.

Impact of noise in machine learning. In traditional TS analysis (Gao et al., 2009), noise is often considered a disruptive factor, prompting widespread use of denoising techniques to reduce noise levels and improve model prediction accuracy and robustness. However, recent research (Zhang, 2007; Nourani and Partoviyan, 2018) has revealed that rather than simply removing noise, injecting noise as a data augmentation strategy can significantly enhance model robustness under certain conditions. Nourani et al. (Nourani and Partoviyan, 2018) demonstrated the potential of noise to increase training data diversity and enhance model performance. Magklaras et al. (Magklaras et al., 2019) suggest that noise injection helps identify malicious data. Kim et al. (Kim and Chung, 2024) integrated noise injection with digital signal processing techniques, using frequency feature extraction to improve the anti-interference ability of classification models in various practical scenarios.

3 Method

We introduce a novel framework, the **N**oise Injection Augmented **LLM** for **TS** (NLTS), which comprises several critical components, including: 1) noise design and sampling strategies for generating diverse noise patterns, 2) data perturbation with noise injection, 3) textualization of a point in TS to convert numerical data of a point into a descriptive textual format, 4) data transformation and tokenization of the full TS to enable prompt formulation and token prediction, and 5) sampling and aggregation approaches to obtain the ultimate TS forecasting. As shown in Fig. 1, we present two approaches for zero-shot TS forecasting using LLMs. Our NLTS (bottom subplot) is purely zero-shot, data-agnostic, and LLM-agnostic, allowing it to seamlessly adapt to various LLMs and tasks.

3.1 TS Forecasting Problem Formulation

Mathematically, a TS $\mathbf{x} = \{x_t\}_{t=1}^{T}$ can be represented as the sum of a signal $\{f(t)\}_{t=1}^{T}$ and noise $\{\epsilon_t\}_{t=1}^{T}$, such that: $x_t = f(t) + \epsilon_t$, where $f(t)$ denotes the true underlying signal at time t, and ϵ_t represents the noise term at time t. In the context of TS forecasting, noise refers to random fluctuations or disturbances within the data that cannot be accounted for by the underlying trend, seasonality, or other systematic patterns. The goal of TS forecasting is to predict the future values $\{x_{T+1}, x_{T+2}, \ldots, x_{T+H}\}$, with H representing the forecast horizon. Thus, the TS forecasting problem can be formulated as estimating the conditional distribution of future values given past observations: $p(\{x_t\}_{t=T+1}^{T+H} | \{x_t\}_{t=1}^{T})$.

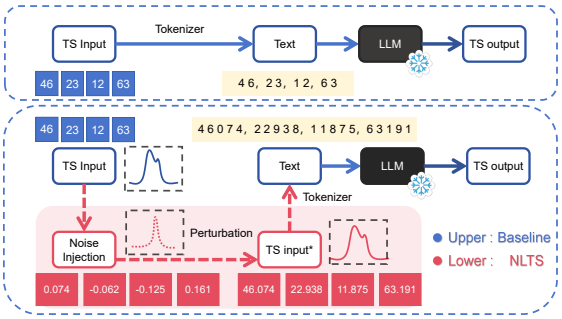


Figure 1: Overview of zero-shot TS forecasting in off-the-shelf LLMs: the top is a vanilla usage of off-the-shelf LLM for TS, where the numerical values are tokenized and directly converted into a string, and then fed into a frozen LLM for prediction. The bottom is our NLTS framework, which introduces noise injection.

3.2 TS Forecasting in Off-the-Shelf LLMs

Token modeling in LLM. LLMs are trained on sequential data, $\mathcal{S} = \{S_1, S_2, \ldots, S_i, \ldots, S_N\}$, where each sequence S_i consists of tokens $(\mathbf{s}_{i,1}, \mathbf{s}_{i,2}, \ldots, \mathbf{s}_{i,j}, \ldots, \mathbf{s}_{i,n_i})$, with each token $\mathbf{s}_{i,j}$ from a vocabulary $\mathcal{V}$. These models encode an autoregressive distribution, where the probability of each token depends only on preceding tokens: $p_\Theta(S_i) = \prod_{j=1}^{n_i} p_\Theta(\mathbf{s}_j \mid \mathbf{s}_{0:j-1})$, and the model parameters Θ are optimized by maximizing the likelihood of the entire dataset: $p_\Theta(\mathcal{S}) = \prod_{i=1}^{N} p_\Theta(S_i)$.

Next-token prediction of TS. Sampling from a trained language model typically begins with an initial prompt $\mathbf{s}_{0:k}$ and progresses iteratively, selecting the subsequent token based on $p_\Theta(\mathbf{s}_j \mid \mathbf{s}_{0:j-1})$. In the case of LLMs, TS forecasting is conceptualized as a next-token generation task. This autoregressive process can be mathematically expressed as: $p(\text{Token}(\tilde{x}_{T+k}) | \{\text{Token}(\tilde{x}_t)\}_{t=1}^{T+k-1})$. Consequently, the conditional distribution is approximated as $p(\tilde{x}_{T+k} | \{\tilde{x}_t\}_{t=1}^{T+k-1}) \approx p(\text{Token}(\tilde{x}_{T+k}) | \{\text{Token}(\tilde{x}_t)\}_{t=1}^{T+k-1})$.

3.3 LLM Forecasting via Noise Injection

Noise design and sampling strategies. The noise term is commonly represented as a random variable with a mean of zero and a variance of σ^2. The variance σ^2 reflects the level of uncertainty or randomness inherent in the observations. For instance, noise that follows a Gaussian distribution is expressed as $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, where ϵ_i denotes the noise to be injected. In addition to the Gaussian

distribution, we examine four other noise distributions, including uniform, Laplace, Gamma, and Beta distributions. A comprehensive discussion of these aspects is provided in Appendix C.1.

We also introduce noise sampling with noise scaling, which entails the controlled adjustment of the noise magnitude or intensity relative to the underlying data, ensuring that the noise does not overwhelm the signal, yet remains sufficiently impactful to affect model behavior and performance. This noise scaling is essential for balancing the model's sensitivity and robustness. Specifically, we parameterize the noise variance σ^2 as $\sigma^2 = \alpha^2\sigma_x^2$, where σ_x^2 is the variance of the original TS, and α is the scaling factor that governs the noise magnitude relative to the original value x_t. The noise is jointly determined by the standard deviation σ_x of the original data and the scaling factor α.

Noise injection on TS. The external noise injection introduces an element of stochasticity into the model's input, effectively simulating data perturbation. Given a TS $x_t \in \mathbb{R}$, *noise injection* is a stochastic perturbation operator $\mathcal{P} : \mathbb{R} \to \mathbb{R}$ that is formally defined as:

$$\mathcal{P}(x_t) = \tilde{x}_t = x_t + \epsilon_i, \tag{1}$$

where ϵ_i is sampled from a noise distribution. This operator induces controlled variability into the input. By enforcing $\tilde{x}_t$ to approximate x_t under $\mathbb{E}[\epsilon_i] = 0$, the model is regularized to prioritize latent signal structures over spurious fluctuations, thereby enhancing generalization without architectural modifications or retraining. The perturbed series $\{\tilde{x}_t\}$ is propagated through subsequent LLM-based forecasting steps, optimizing robustness to potential distributional shifts.

Textualization of point in TS. The textualization of a TS point involves the conversion of quantitative data, such as numerical values, into a natural language format. This process is essential for enabling LLMs to comprehend, interpret, and generate human-readable representations of numerical information. Given a point of TS x_t, its *textualization* is a bijective mapping $\mathcal{T} : \mathbb{R} \to \mathcal{S}$ to a linguistically structured format via: 1) digit separation: $\mathcal{T}_1(x_t) = d_1 \sqcup d_2 \sqcup \cdots \sqcup d_n$, where d_i are decimal digits of x_t, ensuring token-wise independence; 2) precision scaling: For fixed $k \in \mathbb{N}$, the integer representation $\tilde{x}_{t,\text{int}} = \mathcal{T}_2(x_t) = \lfloor x_t \times 10^k \rfloor$, discarding decimals while preserving invertibility via $\mathcal{T}_2^{-1}(\tilde{x}_{t,\text{int}}) = \tilde{x}_{t,\text{int}}/10^k$. The composite function $\mathcal{T}(x_t) = \mathcal{T}_1(\mathcal{T}_2(x_t))$ encodes x into interpretable textual tokens, bridging continuous TS with discrete symbolic frameworks (Gruver et al., 2023).

Tokenization of noised TS. The meticulous tokenization of TS is pivotal in improving the precision and dependability of predictions made by LLMs. Transformer-based LLMs, such as GPT-3.5 and DeepSeek, are inherently designed to process token sequences, typically in textual form, which necessitates the adaptation of TS into a compatible format for these models. Let $\{\tilde{x}_t\}_{t=1}^T \subset \mathbb{R}$ denote a noised TS derived from data perturbation (Eq. (1)). A *tokenization operator* $\mathcal{Q} : \mathbb{R}^T \to \mathcal{S}^T$ is a bijective mapping that converts $\{\tilde{x}_t\}$ into a discrete token sequence $S = \{\text{Token}_t(\tilde{x}_t)\}_{t=1}^T$, where $\mathcal{S}$ is the token vocabulary. The operator satisfies:

$$S = \mathcal{Q}(\{\tilde{x}_t\}) \quad \text{and} \quad \{\tilde{x}_t\} = \mathcal{Q}^{-1}(S), \tag{2}$$

ensuring exact and lossless invertibility between numerical and symbolic representations.

The bijectivity constraints $\mathcal{Q} \circ \mathcal{Q}^{-1} = \text{Id}_{\mathcal{S}^T}$ and $\mathcal{Q}^{-1} \circ \mathcal{Q} = \text{Id}_{\mathbb{R}^T}$ guarantee structural fidelity, where Id denotes identity mapping. Crucially, $\mathcal{Q}$ preserves temporal semantics by aligning token embeddings $\{\text{Token}_t\}$ with the perturbed dynamics $\{\tilde{x}_t\}$, enabling LLMs to process noisy numerical sequences as contextually coherent text. This tokenization-reconstruction duality ensures that stochastic variations introduced during noise injection remain interpretable within the LLM's embedding space. Specifically, we adopt the tokenization method for TS as described in LLM-Time (Gruver et al., 2023). The tokenization function $\mathcal{Q}$ utilizes commas to delineate individual time points, treating each time step as a discrete input, with time steps separated by commas. For example, the TS $[25, 27, 29]$ is encoded as "2 5, 2 7, 2 9". The comma separation is crucial as it allows the model to distinguish between distinct time steps and preserve their sequential order. For instance, by combining textualization and tokenization, the TS $[0.314, 3.14, 31.4, 314.0]$ is represented as "3 1, 3 1 4, 3 1 4 0, 3 1 4 0 0". This sequence S can now be directly input into the LLM.

Sampling and aggregating outputs of LLM. In addition to the insights gleaned from individual model predictions, drawing multiple predictions from an LLM can yield valuable indications regarding the overall central tendency and confidence of LLM-based forecasting. Suppose we intend to approximate the central tendency

of the forecasting function $f(t)$ under a probability distribution $p(t)$ across a domain $\mathcal{X}$. For the testing set $\mathbf{x}_* = \{\tilde{x}_{T+1}, \tilde{x}_{T+2}, \ldots, \tilde{x}_{T+H}\}$, we define $\mathbf{x}_* = \mathcal{Q}^{-1}(\text{Token}(S_*))$, where $S_* = \{\text{Token}_t(\tilde{x}_t)\}_{t=T+1}^{T+H}$. Additionally, we assess the uncertainty of each point via the empirical variance, furnishing a measure of the prediction's reliability. We estimate the central tendency of the prediction by using the sample median. Specifically, we order multiple predictive points in non-decreasing sequence to obtain the order statistics: $x_h^{(1)} \leq x_h^{(2)} \leq \cdots \leq x_h^{(m)}$. The sample median $\bar{x}_*$ is defined as:

$$\bar{x}_* = \begin{cases} x^{(m)}_{*,\frac{h+1}{2}} & \text{if } h \text{ is odd,} \\ \frac{1}{2}\left(x^{(m)}_{*,\frac{h}{2}} + x^{(m)}_{*,\frac{h}{2}+1}\right) & \text{if } h \text{ is even,} \end{cases} \tag{3}$$

where $x^{(m)}_{*,i}$ denotes the i-th point of LLM-generated forecasting, and m is the number of LLM model generations. This aggregation contributes to reducing the effect of individual responses while offering a more robust estimate of the true forecast. In addition, the confidence interval of point-wise prediction can be achieved by leveraging the quantile and variance of m predictions. Finally, we adopt $\bar{\mathbf{x}}_*$ as the overall ultimate TS forecast in LLM. Note that we do not have to eliminate the introduced noise from $\mathbf{x}_*^{(m)}$ or $\bar{\mathbf{x}}_*$, because we believe the LLM has the capacity of noise-based self-correction, adaptively learns to disregard "irrelevant" noise, and concentrates on salient patterns, thereby substantially bolstering its overall predictive performance and forecasting accuracy.

4 Theoretical Analysis of NLTS

The broad and reliable utility of LLMs like GPT best demonstrates their well-trained status because they have converged optimally, residing at a strict local maximum of the empirical log-likelihood.

Theorem 1 (First- and second-order optimality for well-trained LLMs)**.** *Let $\hat{\mathcal{L}}(\Theta) = p_\Theta(\mathcal{S})$ denotes the empirical log-likelihood of an LLM over training datasets. A parameter configuration Θ^* maximizes $\hat{\mathcal{L}}(\Theta)$ and defines a well-trained LLM if: first-order optimality with $\nabla_{\mathbf{w}}\hat{\mathcal{L}}(\Theta^*) = 0$ and second-order optimality with the expected Hessian $\mathbb{E}[H_f(\Theta^*)] = \mathbb{E}[\nabla^2_{\mathbf{w}}\hat{\mathcal{L}}(\Theta^*)]$ is negative definite.*

The negative definiteness of the expected Hessian reflects the intrinsic concavity of the likelihood function in identifiable parameter regimes.

Lemma 1 (Perturbation stability of well-trained LLMs with Gaussian noise)**.** *Let f_Θ be a well-trained LLM satisfying Theorem 1. For an Gaussian perturbed input $\tilde{x}_t$ and target function $h(x)$, the following inequality holds: $\mathbb{E}_\epsilon[f_{\Theta^*}(\tilde{x}_t) - h(x)] \leq \mathbb{E}_\epsilon[f_{\Theta^*}(x) - h(x)]$.*

Lemma 1 indicates that the generalization error of LLM can be reduced by noise injection on the input for zero-shot forecasting. We present the proofs of Theorem 1 and Lemma 1 in Appendix B.

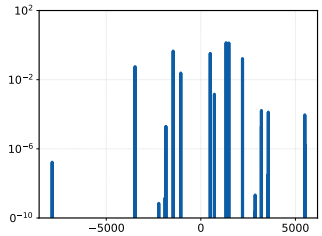


Figure 2: Hessian analysis of GPT-2 Small (125M), estimated via SLQ. The eigenvalues of the Hessian matrix (x-axis) are plotted against their frequency density on a logarithmic scale (y-axis), characterizing the curvature structure of the loss landscape.

Empirical validation: Hessian analysis of GPT-2. To substantiate Theorem 1, we conduct an empirical investigation utilizing the canonical open-source GPT-2 small (125M) architecture (Radford et al., 2019). The parameter scale of GPT-2 renders the full-rank Hessian computations tractable, whereas such analysis is often computationally prohibitive for larger proprietary models. We operate under the premise that the pre-trained GPT-2 model resides in a "well-trained" state, having converged to a stable region of the empirical log-likelihood surface $\hat{\mathcal{L}}(\Theta)$. We adopt the Stochastic Lanczos Quadrature (SLQ) algorithm proposed by (Zhang et al., 2024) to approximate the spectral properties of the global Hessian without the explicit construction of the $125M \times 125M$, which would be infeasible to store. As shown in Fig. 2, we present the global Hessian spectral distribution of the GPT-2 model at convergence, where the calculated trace is approximately -0.0046. At a local optimum where the gradient vanishes, this negative trace serves as a robust indicator of the negative definiteness of the expected Hessian. This finding provides empirical evidence for the intrinsic concavity of the log-likelihood function at the parameter configuration Θ^* necessitated by our theoretical derivation.

5 Zero-Shot Forecasting Experiments

5.1 Setting

In the experiments of NLTS, we partition the dataset into training, validation, and test sets, and select the optimal noise level based on the model's performance on the validation set.

Datasets. We employ three recognized benchmark datasets: Autoformer (Wu et al., 2021), Darts (Herzen et al., 2022), and Memorization (Gruver et al., 2023). The details of these datasets are presented in Appendix E.1.

Models and baselines. We examine nine representative LLMs: GPT-4 (OpenAI, 2023), GPT-3.5-Turbo-Instruct (Brown et al., 2020), Moonshot-V1-8k, Claude-3-Opus, Claude-3.5-Sonnet, Claude-3.5-Haiku (Anthropic, 2024), DeepSeek-V3 (DeepSeek-AI et al., 2024), GLM-4-Air (GLM et al., 2024), and Qwen3-4B (Yang et al., 2025). Observe that we still choose not to employ the newest or more sophisticated LLMs, including higher versions of ChatGPT, owing to their prohibitive cost. Furthermore, the study performs rigorous comparative analysis, contrasting these LLMs' performance with that of non-LLM baseline models, including ARIMA, SM-GP, Temporal Convolution Networks (TCN; Lea et al., 2016), N-BEATS, N-HiTS, and a range of advanced TS forecasting models such as Informer (Zhou et al., 2021), Autoformer, NSTransformer (Liu et al., 2022), TimesNet (Wu et al., 2023), PatchTST (Nie et al., 2023b), and iTransformer (Liu et al., 2023). Additionally, the zero-shot forecasting model LLMTime is included to further highlight the relative effectiveness of LLMs in this domain.

While recent works have proposed several LLM-based forecasting frameworks, such as TEST (Sun et al., 2024), TimeLLM (Jin et al., 2023a), and CALF (Liu et al., 2025), we do not include them as direct baselines, as their so-called zero-shot forecasting involves optimizing the model on one dataset and evaluating it on another. In contrast, we directly apply off-the-shelf LLMs without any task-specific fine-tuning.

Metrics. For performance evaluation of all methods, we employ two widely used regression metrics: Mean Squared Error (MSE) and Mean Absolute Error (MAE). The evaluation setting follows LLMTime (Gruver et al., 2023): Darts typically uses 80% of each TS as prompt input and reserves 20% for testing. The memorization benchmark forecasts the next 30 time steps.

5.2 Main Results

Table 1 and Table 2 report the zero-shot forecasting performances of NLTS on short-term and long-term TS benchmarks, respectively. Based on the optimal noise level selected on the validation set, NLTS demonstrates a clear advantage in both settings, generally achieving the lowest MSE and MAE across the vast majority of datasets, and outperforming all baseline methods. For example, in the short-term IstanbulTraffic dataset, NLTS reduces the MAE by approximately 45% compared to LLMTime; in the more challenging long-term setting (e.g., the ILI dataset), where all methods exhibit high prediction errors, NLTS still achieves the best performance. Notably, LLMTime, which is also a zero-shot method, shows competitive results on multiple tasks. This suggests that the success of LLMs in zero-shot TS forecasting is likely not primarily due to memorizing answers from data contamination, but rather more likely stems from their genuine ability to understand data patterns and make accurate predictions.

5.3 Quantitative Analysis

Impact of noise level. The scale of noise critically determines the ability of LLMs with NLTS to produce accurate forecasts across diverse domains. In this section, we investigate the effect of noise level in LLMs with NLTS. To keep the evaluation focused and informative, we report results on five representative real-world datasets: Traffic, ETTh2, AusBeer, GasRateCO2, and MonthlyMilk. From Fig. 3 (c), we can observe that the forecasting accuracy of GPT-based models with NLTS varies significantly across different noise intensities. Crucially, moderate noise levels, particularly those in the range of 0.005 to 0.02, consistently yield superior and more stable performance. This trend suggests the presence of a noise augmentation "sweet spot", where the benefits of regularization are maximized without introducing excessive distortion. For instance, in Fig. 3 (a) and (b), the model achieves its highest accuracy (60.48%) on the Traffic dataset with an MAE improvement at the noise level of 0.02, demonstrating that a carefully calibrated level of perturbation can effectively enhance model generalization in volatile environments. Similar favorable performance trends are also observed in both the AusBeer and MonthlyMilk datasets.

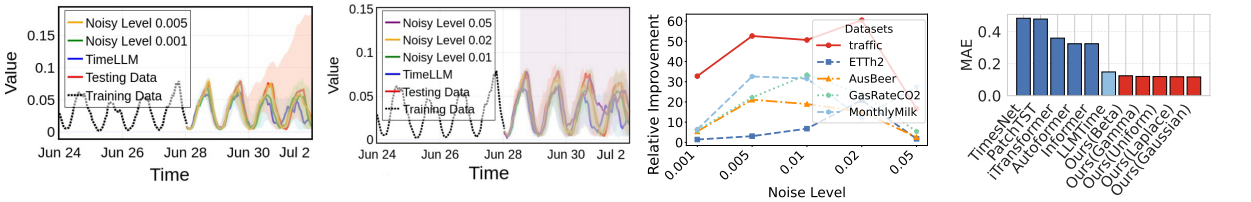
Impact of noise types. Noise type is pivotal to NLTS effectiveness. Different noise types, from

Benchmark	Datasets	H	NLTS		LLMTime		N-HiTS		N-BEATS		TCN		SM-GP		ARIMA	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Darts	AirPassengers	29	**0.003**	**0.044**	0.010	0.075	0.025	0.118	0.028	0.126	0.048	0.172	0.008	0.073	0.006	0.060
	AusBeer	43	**0.001**	**0.020**	0.001	0.026	0.027	0.154	0.006	0.059	0.011	0.094	0.056	0.186	0.002	0.032
	GasRateCO2	60	**0.002**	**0.039**	0.038	0.160	0.052	0.183	0.106	0.271	0.050	0.186	0.033	0.157	0.029	0.151
	HeartRate	180	**0.005**	**0.057**	0.035	0.157	0.114	0.279	0.051	0.174	0.048	0.184	0.063	0.208	0.039	0.162
	MonthlyMilk	34	**0.001**	**0.026**	0.005	0.062	0.006	0.060	0.011	0.088	0.025	0.138	0.014	0.104	0.014	0.108
	Sunspots	141	**0.051**	**0.166**	0.082	0.208	0.070	0.191	0.135	0.287	0.068	0.181	0.088	0.223	0.061	0.176
	Wine	36	**0.008**	**0.067**	0.014	0.086	0.044	0.164	0.046	0.140	0.025	0.127	0.047	0.182	0.012	0.087
	Wooly	24	**0.010**	**0.079**	0.014	0.103	0.012	0.088	0.036	0.178	0.022	0.132	0.031	0.140	0.028	0.156
Memorization	IstanbulTraffic	30	**0.014**	**0.094**	0.136	0.330	0.259	0.401	0.399	0.573	0.122	0.304	0.229	0.385	0.154	0.310
	TSMCStock	30	**0.0003**	**0.014**	0.0004	0.016	0.001	0.018	0.029	0.161	0.002	0.039	0.014	0.108	0.465	0.584
	TurkeyPower	30	**0.001**	**0.023**	0.002	0.032	0.018	0.114	0.019	0.126	0.004	0.046	0.015	0.103	0.003	0.047

Table 1: Performance on *Short-Term Time Series*. **Bold** and underline: the best and the second best performance.

Benchmark	Datasets	H	NLTS		iTransformer		LLMTime		PatchTST		TimesNet		Autoformer		Informer	
			MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Autoformer	ECL	96	**0.010**	**0.076**	0.084	0.210	0.027	0.127	0.124	0.289	0.076	0.208	0.235	0.364	0.124	0.364
	ETTh1	96	**0.019**	**0.108**	0.038	0.163	0.022	0.120	0.071	0.239	0.080	0.262	0.071	0.234	0.039	0.234
	ETTh2	96	**0.028**	**0.134**	0.214	0.359	0.042	0.148	0.331	0.478	0.314	0.483	0.184	0.324	0.174	0.324
	ETTm1	96	**0.002**	**0.035**	0.003	0.044	0.004	0.045	0.006	0.056	0.008	0.069	0.009	0.078	0.004	0.078
	ETTm2	96	**0.009**	**0.077**	0.021	0.117	0.028	0.142	0.026	0.139	0.021	0.124	0.077	0.249	0.075	0.249
	Traffic	96	**0.001**	**0.027**	0.091	0.257	0.010	0.069	0.075	0.227	0.077	0.228	0.073	0.213	0.079	0.213
	ILI	24	**0.016**	**0.109**	0.426	0.486	0.084	0.115	0.979	0.687	0.766	0.765	1.077	0.969	11.047	3.312
Monash	australian_electricity_demand	336	**0.027**	**0.136**	0.344	0.460	0.067	0.196	0.470	0.533	0.344	0.454	0.476	0.547	1.253	0.963
	traffic_hourly	168	**0.002**	**0.032**	0.278	0.355	0.004	0.049	0.265	0.345	0.248	0.327	0.321	0.410	0.951	0.771
	us_births	30	**0.037**	**0.137**	0.201	0.312	0.038	0.141	0.193	0.294	0.172	0.289	0.623	0.609	0.532	0.577

Table 2: Performance on *Long-Term Time Series*. More results of various horizons are provided in Appendix E.5.



(a) NLTS with low ϵ_i ($\alpha < 0.01$) (b) NLTS with high ϵ_i ($\alpha \geq 0.01$) (c) Performance of various ϵ_i (d) MAE (on ETTh2) of various types of ϵ_i

Figure 3: Effect of noise level with NLTS. Subplots (a) and (b) illustrate predictions on the Traffic dataset under low and high noise levels. Subplots (c) and (d) summarize the relative improvements of NLTS over the LLMTime baseline across multiple datasets under varying noise levels and performances of various types of ϵ_i, respectively.

LLMs / α	0.001	0.005	0.010	0.020	0.050
GPT-4	22.82%	-2.19%	11.00%	**29.43%**	3.17%
GPT-3.5	1.41%	3.07%	6.79%	**20.91%**	1.78%
Moonshot-V1-8k	12.49%	9.18%	25.40%	-3.03%	9.76%
Claude-3-Opus	4.69%	13.08%	14.23%	**26.94%**	23.18%
Claude-3.5-Haiku	1.67%	4.55%	23.09%	**25.88%**	22.39%
Claude-3.5-Sonnet	62.71%	**62.97%**	13.72%	-4.02%	15.97%
Deepseek-V3	0.09%	1.90%	0.84%	**3.60%**	3.07%
GLM-4-Air	14.71%	**50.09%**	41.83%	45.05%	27.44%
Qwen3-4B	8.75%	12.56%	10.69%	6.87%	**29.34%**

Table 3: Performance improvement (%) across different LLMs on the ETTh2 dataset achieved through NLTS.

simple unstructured noise to more complex distributions, can affect model dynamics and forecasting performance differently. To assess the effect of noise distribution, we conduct controlled experiments on the ETTh2 dataset with different injected noise types. As shown in Fig. 3(d), Gaussian noise yields the best performance, achieving the lowest MAE of 0.117. Moreover, all noise-augmented NLTS variants outperform both traditional forecasting models and the LLMTime baseline.

Impact of LLM choices. We evaluate a diverse collection of LLMs, covering both open-source and proprietary models, on the ETTh2 dataset using NLTS, without domain-specific fine-tuning. We further emphasize that our focus is on a low-cost, easily deployable off-the-shelf setup that requires no additional pre-training, aiming to facilitate the adoption of LLMs in resource-constrained environments. As shown in Table 3, NLTS yields substantial improvements in forecasting accuracy across most models and noise levels, although the extent and consistency of these gains vary by model. For example, Claude-3.5-Sonnet and GLM-4-Air exhibit the most pronounced enhancements at a low noise level ($\alpha = 0.005$), with relative improvements of **62.97%** and **50.09%**, respectively. Claude-3.5-Haiku and Claude-3-Opus display more stable performance across different noise intensities, each peaking above **25%** at $\alpha = 0.02$. In contrast, GPT-4 and Moonshot-V1-8k present more variability, with fluctuations in performance depending on the noise scale. The consistent improvements observed across diverse LLMs under-

Datasets	Sources	NLTS		iTransformer		LLMTime		TimesNet		NSTransformer		Autoformer		ARIMA	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Synthetic TS	GP (ExpSineSquared)	**0.013**	**0.091**	1.028	0.808	0.016	0.104	0.965	0.769	0.952	0.783	0.946	0.777	0.025	0.123
	GP (Linear)	**0.018**	**0.110**	1.807	1.139	0.019	0.119	1.741	1.109	1.734	1.125	1.752	1.135	0.041	0.171
	GP (Matérn)	**0.012**	**0.083**	1.020	0.754	0.012	0.089	1.013	0.775	1.004	0.771	0.979	0.746	0.022	0.114
	GP (Polynomial)	**0.012**	**0.086**	1.031	0.831	0.016	0.104	1.228	0.914	1.081	0.843	1.117	0.864	0.024	0.122
	GP (RQ)	**0.013**	**0.086**	1.190	0.883	0.014	0.096	1.222	0.892	1.106	0.857	1.168	0.870	0.029	0.140
	GP (RBF)	**0.015**	**0.092**	1.213	0.852	0.017	0.100	1.235	0.855	1.285	0.863	1.222	0.836	0.031	0.138
Latest Stock	DJIAh	**0.0004**	**0.017**	0.039	0.156	0.0006	0.020	0.034	0.142	0.025	0.132	0.137	0.329	0.0016	0.033
	SZ300m	**0.00001**	**0.003**	0.001	0.022	0.00017	0.011	0.001	0.018	0.001	0.028	0.092	0.292	0.00005	0.006
Wireless Traffic	Online users	**0.022**	**0.104**	1.797	0.958	0.025	0.110	1.959	1.056	1.196	0.769	1.322	0.801	0.029	0.130
	Traffic volume	**0.009**	**0.063**	1.201	0.706	0.009	0.064	1.133	0.741	1.096	0.654	1.448	0.770	0.016	0.092
Smart Glasses	Acc_x	**0.004**	**0.050**	0.370	0.472	0.008	0.072	0.846	0.691	2.186	1.180	4.516	1.817	0.004	0.056
	Acc_y	**0.0001**	**0.009**	125.469	9.836	0.0002	0.010	180.371	11.991	276.452	12.440	166.111	11.846	0.0002	0.011
	Acc_z	**0.0001**	**0.008**	4.449	1.770	0.0001	0.009	6.285	2.124	7.495	2.198	4.786	1.918	0.0003	0.015

Table 4: Zero-shot forecasting performances on synthetic and real-world tasks without data contamination.

score NLTS's promise as a lightweight and effective method for enhancing robustness in TS forecasting tasks, with no model-specific adaptation.

5.4 Contamination-free Performance

Considering that millions of information sources are fed into the LLM pre-training and fine-tuning stage, the test data benchmark may be included in the training data of LLM, a phenomenon known as **data contamination** (Dong et al., 2024; Xu et al., 2024). Under data contamination, LLMs superficially perform zero-shot prediction, while in reality merely conduct in-domain prediction. Consequently, traditional benchmarks tend to overestimate the capabilities of LLM-based methods. To address this concern, we design three new benchmarks without data contamination: (1) synthetic data generated by GPs with various kernels, including ExpSineSquared, Linear, Matérn, Polynomial, Rational Quadratic (RQ), and Radial Basis Function (RBF), (2) the latest stock market datasets that are chronologically and substantively disjoint from all evaluated LLMs' training data, including DJIAh (hourly data from May 1 to July 31, 2025) and SZ300m (minute-level data from May 6–9, 2025), (3) private real-world 5G wireless traffic data, including the number of online 5G users and traffic volume, and (4) UESTC-MMEA-CL dataset(Xu et al., 2023) collected from smart glasses (inertial sensor data, 3-axis acceleration) in complex, real-world activity scenarios. More details and experiments can be found in Appendix D.

Given their temporal proximity to the evaluation period, it is highly unlikely that any mainstream LLMs (e.g., GPT-3.5, GPT-4, Claude) had prior exposure to these data, thereby ensuring a rigorous zero-shot evaluation setting. As shown in Table 4, our method, which is based on GPT-3.5-Turbo-Instruct, consistently outperforms all baselines in three datasets. Notably, another LLM-based zero-shot method, LLMTime (Gruver et al., 2023), also achieves competitive performance. This result indicates that the success of LLMs as zero-shot TS forecasters is not necessarily due to memorization of answers caused by data contamination, but is likely due to truly understanding the data pattern and making correct predictions. Moreover, several specialized deep learning methods perform poorly compared to traditional methods such as ARIMA. This finding prompts us to reconsider whether conventional TS forecasting benchmarks have become outdated, and designing more representative benchmarks to accurately evaluate the actual predictive capacity of models is an urgent challenge.

6 Conclusion

We demonstrate that LLMs can function as effective zero-shot forecasters with the simple introduction of noise, negating the need for task-specific fine-tuning. Instead of centering on modifications to the LLMs themselves, our approach underscores strategic noise injection. By harnessing the robust pattern extrapolation capacities of pre-trained LLMs, our method bypasses the substantial time, effort, and domain-specific expertise typically necessary to develop dedicated TS models. It remains especially advantageous in scenarios with constrained data, where conventional training or fine-tuning methods prove to be wholly unworkable due to inadequate information. Across the datasets, noise injection consistently improves predictive accuracy and strengthens the generalization ability of LLMs. Future work should investigate more adaptive noise injection strategies, precisely tailored to specific model characteristics and dataset properties, to further improve the performance.

Limitations

This study employs a prompt-based approach instead of fine-tuning the model. While the use of prompts reduces computational cost and mitigates overfitting, it may not fully harness the potential of the model architecture. Future research could explore integrating fine-tuning with prompt-based methods to further enhance the model's performance in TS forecasting. If someone trades in stocks based on our work, it could result in personal financial risk.

Ethical Considerations

Our work on TS forecasting has proven effective in predicting real data (such as stocks), users making decisions based solely on the results of our work may lead to unknown risks and losses.

References

Samira Abnar, Mostafa Dehghani, Behnam Neyshabur, and Hanie Sedghi. 2021. Exploring the limits of large scale pre-training. *arXiv preprint arXiv:2110.02095*.

Anthropic. 2024. Claude 3.5 sonnet model card addendum. https://www-cdn.anthropic.com/fed9cc193a14b84131812372d8d5857f8f304c52/Model_Card_Claude_3_Addendum.pdf. Accessed: 2024-06-23.

George EP Box and Gwilym M Jenkins. 1968. Some recent advances in forecasting and control. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 17(2):91–109.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, and 1 others. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

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*.

Ching Chang, Wei-Yao Wang, Wen-Chih Peng, and Tien-Fu Chen. 2025. Llm4ts: Aligning pre-trained llms as data-efficient time-series forecasters. *ACM Transactions on Intelligent Systems and Technology*, 16(3):1–20.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports*, 8(1):6085.

Kai Chen, Qinglei Kong, Yijue Dai, Yue Xu, Feng Yin, Lexi Xu, and Shuguang Cui. 2022. Recent advances in data-driven wireless communication using gaussian processes: a comprehensive survey. *China Communications*, 19(1):218–237.

Yijue Dai, Tianjian Zhang, Zhidi Lin, Feng Yin, Sergios Theodoridis, and Shuguang Cui. 2020. An interpretable and sample efficient deep kernel for gaussian process. In *Conference on Uncertainty in Artificial Intelligence*, pages 759–768. PMLR.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, and 1 others. 2024. Deepseek-v3 technical report.

Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. 2024. Generalization or memorization: Data contamination and trustworthy evaluation for large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 12039–12050.

Jianbo Gao, Hussain Sultan, Jing Hu, and Wen-Wen Tung. 2009. Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: a comparison. *IEEE signal processing letters*, 17(3):237–240.

Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Dan Zhang, Diego Rojas, Guanyu Feng, Hanlin Zhao, and 1 others. 2024. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*.

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

Julien Herzen, Francesco Lässig, Samuele Giuliano Piazzetta, Thomas Neuer, Léo Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, and 1 others. 2022. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124):1–6.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and 1 others. 2023a. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*.

Ming Jin, Qingsong Wen, Yuxuan Liang, Chaoli Zhang, Siqiao Xue, Xue Wang, James Zhang, Yi Wang, Haifeng Chen, Xiaoli Li, and 1 others. 2023b. Large models for time series and spatio-temporal data: A survey and outlook. *arXiv preprint arXiv:2310.10196*.

Gyu Il Kim and Kyungyong Chung. 2024. Extraction of features for time series classification using noise injection. *Sensors*, 24(19):6402.

Colin Lea, Rene Vidal, Austin Reiter, and Gregory D Hager. 2016. Temporal convolutional networks: A unified approach to action segmentation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, pages 47–54, Amsterdam, The Netherlands. Springer.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *nature*, 521(7553):436–444.

Haoxin Liu, Zhiyuan Zhao, Jindong Wang, Harshavardhan Kamarthi, and B Aditya Prakash. 2024. Lst-prompt: Large language models as zero-shot time series forecasters by long-short-term prompting. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 7832–7840.

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

Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2023. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*.

Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary transformers: Exploring the stationarity in time series forecasting. *Advances in neural information processing systems*, 35:9881–9893.

Qianli Ma, Zhen Liu, Zhenjing Zheng, Ziyang Huang, Siying Zhu, Zhongzhong Yu, and James T. Kwok. 2024. A survey on time-series pre-trained models. *IEEE Transactions on Knowledge and Data Engineering*, 36(12):7536–7555.

Aristeidis Magklaras, Nikolaos Andriopoulos, and Alexios Birbas. 2019. Noise injection/machine learning fraud detection framework in time series data. In *25th International Conference on Noise and Fluctuations (ICNF 2019)*, Neuchâtel, Switzerland. ICLAB.

Douglas C Montgomery, Cheryl L Jennings, and Murat Kulahci. 2015. *Introduction to time series analysis and forecasting*. John Wiley & Sons, Hoboken, NJ, USA.

Yuqi Nie, Nam H Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023a. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, Kigali Convention Center, Rwanda. OpenReview.net.

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

Vahid Nourani and Afshin Partoviyan. 2018. Hybrid denoising-jittering data pre-processing approach to enhance multi-step-ahead rainfall–runoff modeling. *Stochastic environmental research and risk assessment*, 32(2):545–562.

OpenAI. 2023. Gpt-4 technical report.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, and 1 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Patara Trirat, Yooju Shin, Junhyeok Kang, Youngeun Nam, Jihye Na, Minyoung Bae, Joeun Kim, Byunghyun Kim, and Jae-Gil Lee. 2024. Universal time-series representation learning: A survey. *arXiv preprint arXiv:2401.03717*.

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

Haixu Wu, Jiehui Xu, Jianmin Wang, and Mingsheng Long. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in neural information processing systems*, 34:22419–22430.

Cheng Xu, Shuhao Guan, Derek Greene, M Kechadi, and 1 others. 2024. Benchmark data contamination of large language models: A survey. *arXiv preprint arXiv:2406.04244*.

Linfeng Xu, Qingbo Wu, Lili Pan, Fanman Meng, Hongliang Li, Chiyuan He, Hanxin Wang, Shaoxu Cheng, and Yu Dai. 2023. Towards continual egocentric activity recognition: A multi-modal egocentric activity dataset for continual learning. *IEEE Transactions on Multimedia*, 26:2430–2443.

Linning Xu, Yijue Dai, Jiawei Zhang, Ceyao Zhang, and Feng Yin. 2020. Exact o (n 2) hyper-parameter optimization for gaussian process regression. In *2020 IEEE 30th International Workshop on Machine Learning for Signal Processing (MLSP)*, pages 1–6. IEEE.

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

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, and 1 others. 2025. Qwen3 technical report.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. 2023. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128.

Ceyao Zhang, Tianjian Zhang, Feng Yin, and Abdelhak M Zoubir. 2023. Data-adaptive m-estimators for robust regression via bi-level optimization. *Signal Processing*, 210:109063.

G Peter Zhang. 2007. A neural network ensemble method with jittered training data for time series forecasting. *Information Sciences*, 177(23):5329–5346.

Tianjian Zhang, Feng Yin, and Zhi-Quan Luo. 2021. Fast generic interaction detection for model interpretability and compression. In *International Conference on Learning Representations*.

Yushun Zhang, Congliang Chen, Tian Ding, Ziniu Li, Ruoyu Sun, and Zhi-Quan Luo. 2024. Why transformers need adam: A hessian perspective. *Advances in neural information processing systems*, 37:131786–131823.

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

Tian Zhou, Peisong Niu, Liang Sun, Rong Jin, and 1 others. 2023. One fits all: Power general time series analysis by pretrained lm. *Advances in neural information processing systems*, 36:43322–43355.

Appendix

This appendix provides rigorous technical substantiation and extended empirical analysis to support the methodological and experimental claims in the main text. Structured into five thematic sections, it delivers:

1. AI Assistance Disclosure.

2. Formal Theoretical Foundations: Complete proofs of Theorem 1 and Lemma 1 (Appendix B), ensuring the mathematical validity of our core propositions.

3. Methodological Transparency: Detailed exposition of the NLTS framework with noise-augmented prompts (Appendix C), including noise distributions, perturbation mechanics, prompt design paradigms, and algorithmic implementation.

4. Data Contamination Experiments and Extended Results: Describe the experimental design and datasets used to evaluate the impact without data contamination, and present results on additional datasets, including per-dataset performance and average performance visualizations (see Appendix D).

5. Reproducibility protocols and extended quantitative evidence are provided in Appendix E, including comprehensive experimental specifications on datasets, implementation hyperparameters, and LLM access cost analysis to enable exact replication. In addition, we present granular performance analyses covering multi-LLM benchmarking, horizon sensitivity, noise-level efficacy, and visualizations of forecasting behavior across all evaluated scenarios.

Collectively, these materials fortify the study's academic integrity, offer actionable insights for practitioners, and establish a foundation for further research in time-series augmentation strategies.

A AI Assistance Disclosure

In the preparation of this paper, we utilized LLM-based AI assistants (e.g., ChatGPT) exclusively for linguistic refinement, including grammar correction, stylistic polishing, and phrasing improvements. These tools were not employed for generating research ideas, designing experiments, analyzing data, interpreting results, or drafting the core scientific contributions. We affirm that all substantive intellectual content, experimental designs, analytical methods, and conclusions presented in this work are entirely our own, and we assume full responsibility for the accuracy and integrity of the paper.

B Proof of Theorem

B.1 Proof of theorem 1

Step 1: Let $\hat{\mathcal{L}}(\Theta) = \log p_\Theta(\mathcal{S})$ denote the empirical log-likelihood of an LLM over a training dataset $\mathcal{S}$, where $p_\Theta(\mathcal{S})$ is the probability assigned to $\mathcal{S}$ under parameters Θ. By definition, Θ^* maximizes $\hat{\mathcal{L}}(\Theta)$ if $\hat{\mathcal{L}}(\Theta^*) \geq \hat{\mathcal{L}}(\Theta)$ for all Θ in a neighborhood of Θ^*. A necessary condition for this is that the gradient vanishes:

$$\nabla_\Theta \hat{\mathcal{L}}(\Theta^*) = \mathbb{E}_{S\sim\mathcal{S}}\left[\nabla_\Theta \log p_\Theta(\mathcal{S})\right]\Big|_{\Theta=\Theta^*} = 0. \tag{4}$$

This follows from Fermat's theorem in optimization: extrema of differentiable functions occur at critical points where the gradient is zero.

Step 2: To ensure Θ^* is a local maximum, we examine the second-order Taylor expansion of $\hat{\mathcal{L}}(\Theta)$ around Θ^*:

$$\hat{\mathcal{L}}(\Theta^* + \Delta\Theta) = \hat{\mathcal{L}}(\Theta^*) + \Delta\Theta^\top \nabla_\Theta \hat{\mathcal{L}}(\Theta^*) + \frac{1}{2}\Delta\Theta^\top \nabla^2_\Theta \hat{\mathcal{L}}(\Theta^*)\Delta\Theta + o(\|\Delta\Theta\|^2). \tag{5}$$

Substituting $\nabla_\Theta \hat{\mathcal{L}}(\Theta^*) = 0$, the dominant term is the quadratic form $\frac{1}{2}\Delta\Theta^\top H_f(\Theta^*)\Delta\Theta$. For Θ^* to be a local maximum, this term must be negative for all $\Delta\Theta \neq 0$, which requires $H_f(\Theta^*) \prec 0$.

Step 3: The empirical Hessian $H_f(\Theta^*) = \nabla^2_\Theta \hat{\mathcal{L}}(\Theta^*)$ is evaluated on the training set $\mathcal{S}$. To generalize beyond $\mathcal{S}$, consider the expected Hessian over the data distribution $\mathcal{D}_S$:

$$\mathbb{E}[H_f(\Theta^*)] = \mathbb{E}_{S\sim\mathcal{D}_S}\left[\nabla^2_\Theta \log p_\Theta(\mathcal{S})\right]\Big|_{\Theta=\Theta^*}. \tag{6}$$

By the law of large numbers, $H_f(\Theta^*) \to \mathbb{E}[H_f(\Theta^*)]$ as $|\mathcal{S}| \to \infty$. Negative definiteness of $\mathbb{E}[H_f(\Theta^*)]$ ensures the curvature remains concave in expectation, preventing overfitting to $\mathcal{S}$.

Step 4: The Fisher information matrix $F(\Theta^*)$ is defined as:

$$F(\Theta^*) = \mathbb{E}_{S\sim\mathcal{D}_S}\left[\nabla_\Theta \log p_\Theta(\mathcal{S}) \nabla_\Theta \log p_\Theta(\mathcal{S})^\top\right]\Big|_{\Theta=\Theta^*}. \tag{7}$$

Under regularity conditions, the expected Hessian satisfies:

$$\mathbb{E}[H_f(\Theta^*)] = -F(\Theta^*). \tag{8}$$

Since $F(\Theta^*) \succ 0$ for identifiable models, $\mathbb{E}[H_f(\Theta^*)] \prec 0$, confirming negative definiteness.

Step 5: In over-parameterized LLMs, the Hessian $H_f(\Theta^*)$ has a high-dimensional nullspace but satisfies $\mathbb{E}[H_f(\Theta^*)] \prec 0$ in non-degenerate directions. This ensures that, despite non-convexity, the model converges to a flatter minimum where the dominant curvature is concave, aligning with empirical observations of robust generalization.

Therefore, a well-trained LLM satisfies $\nabla_\Theta \hat{\mathcal{L}}(\Theta^*) = 0$ and $\mathbb{E}[H_f(\Theta^*)] \prec 0$. These conditions jointly certify that Θ^* is a strict local maximum of the empirical log-likelihood, with stable curvature properties that generalize beyond the training set.

B.2 Proof of Lemma 1 (Perturbation stability with Gaussian noise)

Let $f_{\Theta^*} : \mathcal{X} \to \mathcal{Y}$ be a well-trained LLM satisfying the first- and second-order optimality conditions in Theorem 1, and let $h : \mathcal{X} \to \mathcal{Y}$ be a target function.

Step 1: Let $\tilde{x}_t = x + \alpha\epsilon$, where α scales the noise magnitude. We expand $f_{\Theta^*}(\tilde{x}_t)$ around x by using a second-order Taylor series:

$$f_{\Theta^*}(\tilde{x}_t) = f_{\Theta^*}(x) + \alpha\epsilon^\top \nabla_x f_{\Theta^*}(x) + \frac{\alpha^2}{2}\epsilon^\top H_x(f_{\Theta^*})\epsilon + o(\alpha^2), \tag{9}$$

where $H_x(f_{\Theta^*}) = \nabla_x^2 f_{\Theta^*}(x)$ is the input-space Hessian of the LLM.

Step 2: We take expectations over $\epsilon \sim \mathcal{N}(0, \sigma^2)$:

$$\mathbb{E}_\epsilon[f_{\Theta^*}(\tilde{x}_t)] = f_{\Theta^*}(x) + \frac{\alpha^2\sigma^2}{2}\mathrm{Tr}(H_x(f_{\Theta^*})) + o(\alpha^2), \tag{10}$$

since $\mathbb{E}[\epsilon] = 0$ and $\mathbb{E}[\epsilon^\top H_x(f_{\Theta^*})\epsilon] = \sigma^2\mathrm{Tr}(H_x(f_{\Theta^*}))$.

Step 3: We subtract $h(x)$ from both sides:

$$\mathbb{E}_\epsilon[f_{\Theta^*}(\tilde{x}_t) - h(x)] = \underbrace{f_{\Theta^*}(x) - h(x)}_{\text{Baseline Error}} + \frac{\alpha^2\sigma^2}{2}\mathrm{Tr}(H_x(f_{\Theta^*})) + o(\alpha^2). \tag{11}$$

The inequality $\mathbb{E}_\epsilon[f_{\Theta^*}(\tilde{x}_t) - h(x)] \le \mathbb{E}_\epsilon[f_{\Theta^*}(x) - h(x)]$ reduces to:

$$\frac{\alpha^2\sigma^2}{2}\mathrm{Tr}(H_x(f_{\Theta^*})) \le 0. \tag{12}$$

Step 4: From Theorem 1, the parameter-space Hessian $\mathbb{E}[H_f(\Theta^*)] = \mathbb{E}[\nabla_\Theta^2 \hat{\mathcal{L}}(\Theta^*)] \prec 0$. By the chain rule, the input-space Hessian relates to the parameter-space Hessian via:

$$H_x(f_{\Theta^*}) = J_x(\Theta^*)^\top \mathbb{E}[H_f(\Theta^*)] J_x(\Theta^*), \tag{13}$$

where $J_x(\Theta^*) = \nabla_\Theta f_{\Theta^*}(x)$ is the Jacobian of f_{Θ^*} with respect to Θ. Since $\mathbb{E}[H_f(\Theta^*)] \prec 0$, it follows that $H_x(f_{\Theta^*}) \preceq 0$, and thus $\mathrm{Tr}(H_x(f_{\Theta^*})) \le 0$.

Therefore, the inequality demonstrates that input perturbations reduce the expected error when the LLM is well-trained. This aligns with Theorem 1's second-order condition, which enforces flatness in the loss landscape, making the model resilient to input variations. The lemma formalizes how noise injection regularizes LLM predictions by exploiting the concave curvature guaranteed by Theorem 1.

B.3 Perturbation stability with Laplace noise

Lemma 2 (Perturbation stability of well-trained LLMs with Laplace noise)**.** *Let f_{Θ^*} be a well-trained LLM satisfying Theorem 1. For any perturbed input $\tilde{x}_t = x + \alpha\epsilon$, where ϵ is an independent Laplace noise with location $\mu = 0$ and scale $b > 0$, and for any target function $h(x) : \mathcal{X} \to \mathcal{Y}$, the following inequality holds:* $\mathbb{E}_\epsilon\left[f_{\Theta^*}(\tilde{x}_t) - h(x)\right] \le \mathbb{E}_\epsilon\left[f_{\Theta^*}(x) - h(x)\right]$.

The proof follows the same steps as Lemma 1, replacing the Gaussian moments with those of the Laplace distribution: $\mathbb{E}[\epsilon_i] = 0$, $\mathbb{E}[\epsilon_i^2] = 2b^2$, and $\mathbb{E}[\epsilon_i\epsilon_j] = 0$ for $i \neq j$. The second-order Taylor expansion yields an extra term $\alpha^2 b^2\,\mathrm{Tr}(H_x(f_{\Theta^*}))$, and the negative definiteness of the expected Hessian from Theorem 1 ensures $\mathrm{Tr}(H_x(f_{\Theta^*})) \le 0$, establishing the inequality. Specifically, let f_{Θ^*} be a well-trained LLM satisfying Theorem 1. For any

perturbed input $\tilde{x}_t = x + \alpha\epsilon$, where ϵ is an independent Laplace random variables with location 0 and scale b, and for a target function $h(x)$, we have the following proof.

Step 1: By leveraging the Taylor expansion, we expand $f_{\Theta^*}(x + \alpha\epsilon)$ around x to second order:

$$f_{\Theta^*}(x+\alpha\epsilon) = f_{\Theta^*}(x) + \alpha\epsilon^\top \nabla_x f_{\Theta^*}(x) + \frac{\alpha^2}{2}\epsilon^\top H_x(f_{\Theta^*})\epsilon + o(\alpha^2), \tag{14}$$

where $H_x(f_{\Theta^*}) = \nabla_x^2 f_{\Theta^*}(x)$.

Step 2: Since the Laplace distribution is symmetric with mean zero, $\mathbb{E}[\epsilon] = 0$. For independent components, $\mathbb{E}[\epsilon_i\epsilon_j] = 0$ for $i \neq j$, and $\mathbb{E}[\epsilon_i^2] = 2b^2$. Thus,

$$\begin{aligned}\mathbb{E}[\epsilon^\top H_x(f_{\Theta^*})\epsilon] &= \sum_{i,j} H_{ij}\mathbb{E}[\epsilon_i\epsilon_j] \\ &= \sum_i H_{ii}\mathbb{E}[\epsilon_i^2] \\ &= 2b^2 \operatorname{Tr}(H_x(f_{\Theta^*})).\end{aligned} \tag{15}$$

Taking the expectation of the expansion:

$$\mathbb{E}_\epsilon[f_{\Theta^*}(x+\alpha\epsilon)] = f_{\Theta^*}(x) + \alpha^2 b^2 \operatorname{Tr}(H_x(f_{\Theta^*})) + o(\alpha^2). \tag{16}$$

Step 3: For the error difference, we subtract $h(x)$:

$$\mathbb{E}_\epsilon[f_{\Theta^*}(x+\alpha\epsilon) - h(x)] = [f_{\Theta^*}(x) - h(x)] + \alpha^2 b^2 \operatorname{Tr}(H_x(f_{\Theta^*})) + o(\alpha^2). \tag{17}$$

Therefore, the desired inequality reduces to

$$[f_{\Theta^*}(x) - h(x)] + \alpha^2 b^2 \operatorname{Tr}(H_x(f_{\Theta^*})) + o(\alpha^2) \leq [f_{\Theta^*}(x) - h(x)], \tag{18}$$

which, for small α, implies

$$\alpha^2 b^2 \operatorname{Tr}(H_x(f_{\Theta^*})) \leq 0. \tag{19}$$

Step 4: From Theorem 1, the expected parameter-space Hessian $\mathbb{E}[H_f(\Theta^*)] = \mathbb{E}[\nabla_\Theta^2 \hat{\mathcal{L}}(\Theta^*)]$ is negative definite. To relate the input-space Hessian to the parameter-space Hessian, we consider the chain rule,

$$H_x(f_{\Theta^*}) = J_x(\Theta^*)^\top \mathbb{E}[H_f(\Theta^*)] J_x(\Theta^*), \tag{20}$$

where $J_x(\Theta^*) = \nabla_\Theta f_{\Theta^*}(x)$. Negative definiteness of $\mathbb{E}[H_f(\Theta^*)]$ implies $H_x(f_{\Theta^*})$ is negative semidefinite, so $\operatorname{Tr}(H_x(f_{\Theta^*})) \leq 0$. Hence, the inequality holds.

Remark 2. The proof differs from the Gaussian case only in the constant factor (b^2 instead of $\sigma^2/2$), but the sign condition remains unchanged, confirming that Laplace noise injection also reduces expected error for a well-trained LLM.

B.4 Empirical validation: Hessian analysis of GPT-2.

To substantiate the theoretical framework established in Theorem 1, we conduct an empirical investigation utilizing GPT-2 with 125M parameters(Radford et al., 2019). GPT-2 represents an ideal testbed for this analysis because it is a canonical open-source Transformer. The parameter scale of GPT-2 renders the full-rank Hessian computations tractable, whereas such analysis is often computationally prohibitive for larger proprietary models. This allows for high-fidelity local deployment and granular curvature analysis within a constrained computational environment. We adopt the Stochastic Lanczos Quadrature (SLQ) algorithm proposed by Zhang et al. (2024). To further probe the internal dynamics, we compute the Jensen-Shannon distance (JSD) between the Hessian spectral distributions of individual layers (Fig. 4). Our analysis uncovers significant block heterogeneity, particularly between the embedding layer and Transformer blocks. The high JS distances indicate that the optimization landscape is highly non-uniform across different functional components, reflecting a substantial divergence in curvature throughout the model's depth.

C Design of NLTS with Noise-Augmented Prompts

C.1 Noise distribution

The Gaussian distribution is defined by two parameters: the mean μ and the variance σ^2. The probability density function (PDF) of the Gaussian distribution is given by:

$$f(\epsilon_i|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(\epsilon_i-\mu)^2}{2\sigma^2}}. \tag{21}$$

The Uniform distribution on the interval $[a, b]$ is a continuous distribution where every value within

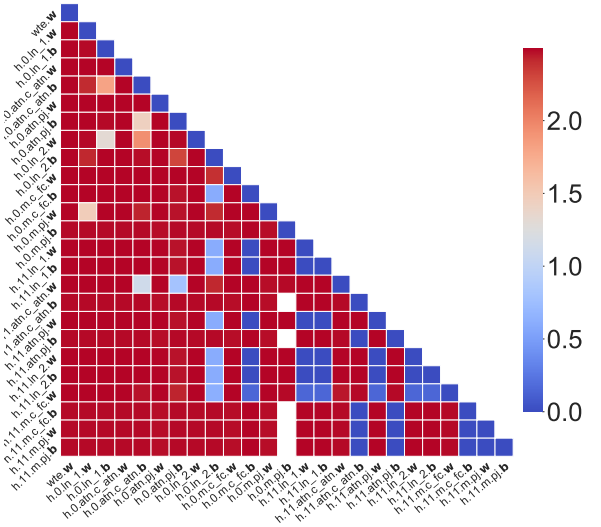


Figure 4: JSD between Hessian spectra of layers. The JSD captures the similarity between Hessian distributions across different layers, where lower values (blue) indicate a higher degree of similarity in the local curvature.

the interval is equally likely. The PDF is:

$$f(\epsilon_i|a,b) = \frac{1}{b-a} \quad \text{for } a \leq \epsilon_i \leq b. \tag{22}$$

The uniform distribution serves as a reference for other distributions and is often used in simulations and modeling when no prior information about the distribution of the noise is available. The Laplace distribution, also known as the double exponential distribution, has a sharp peak at its mean and heavier tails compared to the normal distribution. The PDF of a Laplace distributed random variable ϵ_i with location parameter μ and scale parameter b is:

$$f(\epsilon_i|\mu,b) = \frac{1}{2b} e^{-\frac{|\epsilon_i-\mu|}{b}}. \tag{23}$$

The Gamma distribution is a two-parameter family of continuous probability distributions defined by shape parameter k and scale parameter θ. The PDF of a Gamma distributed random variable ϵ_i is:

$$f(\epsilon_i|k,\theta) = \frac{\epsilon_i^{k-1} e^{-\frac{\epsilon_i}{\theta}}}{\theta^k \Gamma(k)} \quad \text{for } \epsilon_i > 0. \tag{24}$$

The Beta distribution is defined on the interval $[0,1]$ and is characterized by two shape parameters, α and β. The PDF of a beta-distributed random variable ϵ_i is:

$$f(\epsilon_i|\alpha,\beta) = \frac{\epsilon_i^{\alpha-1}(1-\epsilon_i)^{\beta-1}}{B(\alpha,\beta)}. \tag{25}$$

where $B(\alpha,\beta)$ is the beta function.

C.2 Distribution of data perturbation with noise injection

In this section, we evaluate the impact of noise injection on the distributional alignment of LLM-based time series forecasting methods across two benchmark datasets: Traffic and ETTh2. Input data distributions are analyzed before and after noise injection (Figures 6 (a), 6 (d)), revealing how stochastic perturbations modify the training data while preserving underlying temporal patterns. The core comparison involves two methodologies: LLMTime (a baseline without noise injection) and NLTS. Test set outputs are evaluated across three LLM architectures—GPT-3.5-Turbo-Instruct, GLM-Air, and Claude-3.5-Sonnet—to assess generalization.

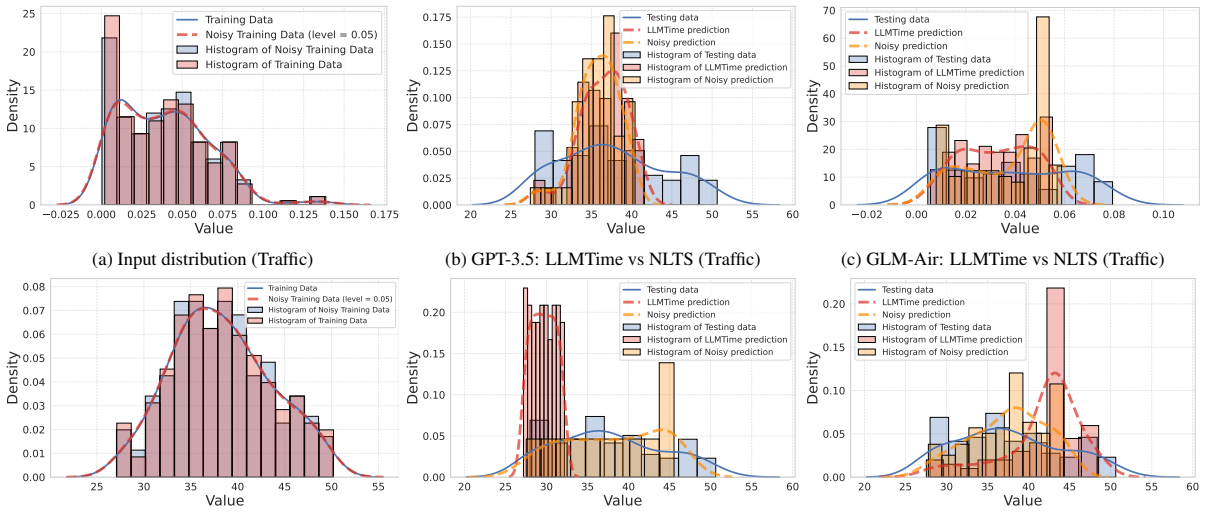
Results demonstrate that NLTS outputs exhibit significantly closer alignment with the true data distribution compared to LLMTime (Figures 6 (b), (c), (e), and (f)). This alignment is consistent across both datasets and all three LLMs, indicating that noise injection enhances robustness by reducing overfitting to training idiosyncrasies. The improved distributional fidelity suggests that NLTS mitigates the domain gap between training and testing phases, enabling better adaptation to real-world variability. The experiment underscores the critical role of structured noise in stabilizing LLM-based forecasts and highlights the framework's capacity to generalize across heterogeneous architectures and datasets.

C.3 Prompt example for TS forecasting

In this experiment, the model's task is to perform time series forecasting based on a sequence of historical observations and generate future predictions. Using the MonthlyMilk dataset as an example, we employ two different prompt strategies.

The **first strategy** is a *raw numeric sequence prompt*, in which historical observations undergo textualization, and then directly provided to the model as the prompt for forecasting. In its noisy variant, controlled levels of noise (α) are first injected into the original numerical observations, which are then textualized in the same manner and provided to the model as the prompt.

The **second strategy** is a *structured chat prompt*, where the input is formatted as a dialogue between the system and the user. The system message defines the forecasting task and constraints, while the user message provides the historical sequence and requests the model to continue it. As with the



(a) Input distribution (Traffic) (b) GPT-3.5: LLMTime vs NLTS (Traffic) (c) GLM-Air: LLMTime vs NLTS (Traffic)

(d) Input distribution (ETTh2) (e) Claude: LLMTime vs NLTS (ETTh2) (f) GLM-Air: LLMTime vs NLTS (ETTh2)

Figure 5: Distribution comparison on the **Traffic** and **ETTh2** dataset. Figures (a) and (d) show the empirical distributions of the training data before and after noise injection. Figures (b)-(c) illustrate test set output distributions on Traffic using LLMTime (without noise) and NLTS (with noise), across GPT-3.5-Turbo-Instruct and GLM-Air. Figures (e)-(f) show results on ETTh2 across Claude-3.5-Sonnet and GLM-Air. In all cases, NLTS outputs align more closely with the true data distribution, highlighting the robustness improvements from noise injection.

numeric prompts, noisy variants are also used.

Raw numeric sequence prompt

```
# Raw numeric sequence prompt:
"5 1 8 , 4 7 3 , 6 0 0 , 6 2 6 , 7 4 1 , 6 9 2
     , 6 0 0 , 5 3 4 , 4 8 4 , 4 9 9 , 4 6 0 ,
     5 0 7 , 5 3 6 , 4 8 1 , 6 2 1 , 6 5 4 , 7
     6 5 , 7 2 3 , 6 3 3 , 5 6 3 , 5 0 9 , 5 1
     5 , 4 8 0 , 5 3 3 ..."

# Noisy prompt with noise level $\alpha = 0.005
     $:
"5 2 0 , 4 7 1 , 5 9 9 , 6 2 6 , 7 4 0 , 6 9 1
     , 6 0 3 , 5 3 5 , 4 8 3 , 4 9 8 , 4 6 1 ,
     5 0 5 , 5 3 4 , 4 7 9 , 6 2 1 , 6 5 7 , 7
     6 7 , 7 2 6 , 6 3 4 , 5 6 2 , 5 0 8 , 5 1
     7 , 4 7 7 , 5 3 2 ..."

# Noisy prompt with noise level $\alpha = 0.05$
     :
"5 2 7 , 4 8 4 , 6 1 6 , 6 3 4 , 7 4 3 , 6 9 0
     , 6 0 8 , 5 2 8 , 4 9 8 , 5 0 6 , 4 5 9 ,
     5 1 0 , 5 3 8 , 4 8 4 , 6 1 7 , 6 7 0 , 7
     7 0 , 7 2 8 , 6 3 3 , 5 6 9 , 5 1 1 , 5 2
     6 , 4 7 6 , 5 2 5 ..."
```

Structured chat prompt

```
# Structured Chat Prompt Example
[
  {
    "role": "system",
    "content": "You are a helpful assistant
         specialized in time series forecasting
         . The user provides a comma-separated
         sequence of decimal numbers, and you
         will predict the following values."
  },
  {
    "role": "user",
    "content": "Please continue the sequence
         without any additional text or
         explanation. Only output the predicted
          numbers.\nSequence:\n5 1 8 , 4 7 3 ,
         6 0 0 , 6 2 6 , 7 4 1 , 6 9 2 , 6 0 0
         , 5 3 4 , 4 8 4 , 4 9 9 , 4 6 0 , 5 0
         7 , 5 3 6 , 4 8 1 , 6 2 1 , 6 5 4 , 7
         6 5 , 7 2 3 , 6 3 3 , 5 6 3 , 5 0 9 ,
         5 1 5 , 4 8 0 , 5 3 3 ..."
  }
]
# Corresponding noisy input example
[
  {
    "role": "user",
    "content": "Please continue the sequence
         without any additional text or
         explanation. Only output the predicted
          numbers.\nSequence:\n5 2 0 , 4 7 1 ,
         5 9 9 , 6 2 6 , 7 4 0 , 6 9 1 , 6 0 3
         , 5 3 5 , 4 8 3 , 4 9 8 , 4 6 1 , 5 0
         5 , 5 3 4 , 4 7 9 , 6 2 1 , 6 5 7 , 7
         6 7 , 7 2 6 , 6 3 4 , 5 6 2 , 5 0 8 ,
         5 1 7 , 4 7 7 , 5 3 2 ..."
  }
]
```

C.4 Algorithm of NLTS

In Algorithm. 1, we establish a holistic methodology for improving TS forecasting performance using LLMs.

Algorithm 1 NLTS for TS forecasting

Require: A TS $\{x_t\}_{t=1}^{T}$, scaling factor α, tokenization function $\mathcal{Q}$;
Ensure: $\bar{\mathbf{x}}_{*,i}$, $\mathrm{Var}(\bar{\mathbf{x}}_{*,i})$;

1: Noise design and sampling;
2: Perturbation of TS by noise injection;
3: Textualizing the points within TS;
4: Tokenize the noised TS in Eq. (2);
5: Provide noisy Tokens as prompts to LLM;
6: Predict tokens for testing points;
7: Invert predicted Tokens $\mathbf{x}_* = \mathcal{Q}^{-1}(\mathrm{Token}(S_*))$;
8: Calculate $\bar{\mathbf{x}}_{*,i}$, $\mathrm{Var}(\bar{\mathbf{x}}_{*,i})$ to aggregate outputs from LLM.

D Impact of Data Contamination

D.1 Dataset

Synthetic TS. We employ multiple Gaussian Process kernel functions, including **ExpSineSquared, Linear, Matérn, Polynomial, Rational Quadratic (RQ), and Radial Basis Function (RBF)**, to generate **1,000 univariate time series** for each kernel. The last **30** time steps of each series are held out as the forecasting target. For benchmarking, we compare our method against state-of-the-art models such as Autoformer, NSTransformer, TimesNet, and iTransformer, the classical ARIMA model, and zero-shot models like LLMTime. All models are configured following the settings described in Appendix E.2

Latest Stock TS. We collected stock price data and constructed **eight real-world** time series datasets covering multiple stock indices across different countries. For the DJIA and SPX indices, the datasets contain 434 points at hourly frequency and 1612 points at 15-minute frequency. The HS300 index contains hourly and minute-level data with 248 and 992 points, while the SZ300 index comprises hourly and minute-level data with 100 and 960 points, respectively.

All datasets are collected **after April 2025**, ensuring that mainstream LLMs (e.g., GPT-3.5, GPT-4, Claude) are unlikely to have seen them during pretraining. In our experimental setup, the last **seven time steps** of each series are reserved as forecasting targets, allowing us to evaluate model performance in short-term prediction scenarios. For benchmarking, we adopt the same set of representative baselines as in the Synthetic TS experiments, and the experimental results are reported in Table 5.

Private Wireless Traffic. The **wireless communication dataset** is private and collected from a 5G base station. Wireless traffic prediction has been a long-standing demand for wireless network planning and management. There are many traffic-related issues in wireless communication suitable for an LLM model, such as wireless traffic analysis, cellular traffic load prediction, traffic load balancing for multimedia multipath systems, channel prediction for communication- relay UAV, and stochastic link modeling of static wireless sensor networks. Highly accurate wireless traffic prediction can reduce the uncertainty of network load and reflect the traffic behavior in the wireless network, which greatly matches the benefits of LLM. The dataset was collected in a southern city in China in 2021. There are multiple patterns with different time scales in the varying of the number of online 5G users, such as long-term, short-term, and mid-term trends. The last **96** time steps of each series are held out as the forecasting target. For benchmarking, we compare our method against state-of-the-art models and LLMTime. All models are configured following the settings described in Appendix E.2

Smart Glasses. We use the UESTC-MMEA-CL dataset(Xu et al., 2023), which contains high-frequency multimodal sensor data collected from smart glasses in complex, real-world activity scenarios. For our experiments, we focus on the raw inertial sensor data, specifically the 3-axis acceleration, which are pre-processed by dividing the raw sensing values by the sensitivity factor ($R_{acc} = 16384$. To ensure the statistical reliability of our findings, we conduct the evaluation on specific samples (e.g. '3_drinking_2020_12_01_16_46_42').

D.2 Average forecasting performance of LLMs

For contamination-free evaluation, we assess the Synthetic and Latest Stock TS datasets, which contain 6 and 8 sub-datasets. Table 6 and Fig. 6 report the average zero-shot forecasting results on both the Synthetic and Latest Stock TS datasets. Overall, our proposed NLTS model consistently achieves the lowest MSE and MAE across benchmarks, demonstrating superior forecasting capability. Since both benchmarks are constructed to eliminate data contamination, the results further in-

Datasets	NLTS		iTransformer		LLMTime		TimesNet		NSTransformer		Autoformer		ARIMA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DJIAh	**0.0004**	**0.017**	0.039	0.156	0.0006	0.020	0.034	0.142	0.025	0.132	0.137	0.329	0.0016	0.033
DJIAm	**0.0002**	**0.009**	0.020	0.121	0.0007	0.022	0.009	0.069	0.028	0.145	0.034	0.169	0.0004	0.017
SPXh	**0.0002**	**0.013**	0.012	0.082	0.0002	0.013	0.016	0.093	0.012	0.084	0.027	0.116	0.0007	0.020
SPXm	0.0002	0.011	0.010	0.086	0.1182	0.142	0.012	0.093	0.015	0.098	0.008	0.073	**0.0001**	**0.010**
SZ300h	**0.00031**	**0.016**	0.046	0.187	0.00075	0.023	0.012	0.084	0.014	0.106	0.146	0.345	0.00082	0.023
SZ300m	**0.00001**	**0.003**	0.001	0.022	0.00017	0.011	0.001	0.018	0.001	0.028	0.092	0.292	0.00005	0.006
HS300h	**0.022**	**0.089**	1.276	0.950	0.120	0.188	1.385	0.968	0.923	0.821	0.878	0.794	0.054	0.199
HS300m	**0.0005**	**0.018**	0.070	0.228	0.0011	0.030	0.044	0.185	0.029	0.139	0.147	0.347	0.001	0.026

Table 5: Forecasting performance (MSE / MAE) on Latest Stock TS. **Bold** and underline: the best and the second best results.

dicate that the superior performance of LLM-based approaches (such as ours and LLMTime) does not stem from memorization of the test data, but rather reflects the LLM's genuine ability to model time series patterns in a zero-shot setting. Traditional methods, such as ARIMA, achieve relatively good results on real stock datasets, whereas some specialized deep learning methods perform worse in the same tasks. These findings prompt us to reconsider whether conventional time series forecasting benchmarks have become outdated and highlight the need to design more representative benchmarks to accurately evaluate the actual predictive capabilities of models.

E Experiment Details and Analysis

E.1 Dataset

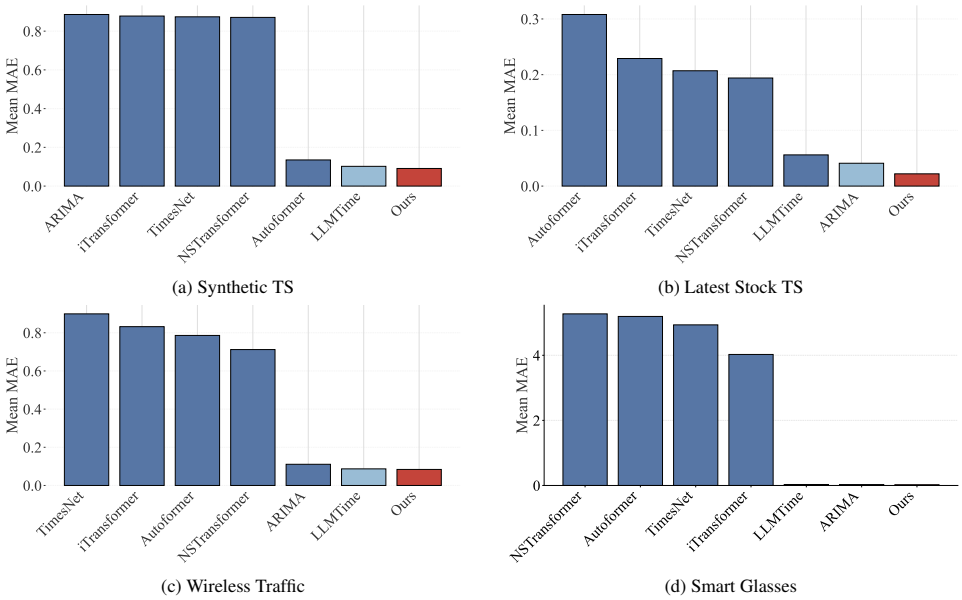
Darts. This dataset comprises eight real univariate time series, including **AirPassengers, AusBeer, GasRateCO2, MonthlyMilk, Sunspots, Wine, Wooly, and HeartRate**. These time series are relatively short, each containing **no more than 800** observations. The first 80% of each time series is used **as prompt input** for the LLMs to generate forecasts, while the last 20% serves as the test set for evaluation. We assess the performance of several methods from the Darts package, including neural network models such as Temporal Convolutional Networks (TCN) , N-BEATS , and N-HiTS , as well as traditional statistical approaches like ARIMA , and the Spectral Mixture Gaussian Process (SM-GP), a Bayesian nonparametric method. Darts serves as an invaluable tool for benchmarking time series forecasting models, making it particularly well-suited for evaluating the performance of LLMTime.

Memorization. This dataset includes three time series sourced from Kaggle: **Istanbul Traffic, TSMC Stock, and Turkey Power.** The Istanbul Traffic time series provides minute-by-minute traffic index data for Istanbul from October 2022 to May 2023. We selected the "TI" column and downsampled the data to an hourly frequency for the period from May 5 to May 18, 2023, resulting in **267 observations**. The TSMC Stock contains daily stock market trading data for Taiwan Semiconductor Manufacturing Company (TSMC) for 2022. We used the closing price column, which consists of **246 observations**. The Turkey Power includes hourly electricity generation and consumption data for Turkey from January 1, 2020, to December 31, 2022. We selected the "Total" column and downsampled the data for 2022 to a daily frequency, resulting in **366 observations**. These time series are short, each containing **no more than 400** observations. The last 30 observations from each time series are reserved for testing. We evaluate the Memorization dataset using the same models as those applied to the Darts dataset.

Autoformer. This dataset consists of nine widely used multivariate TS benchmarks. For our experiments, we select **ETTh1, ETTh2, ETTm1, ETTm2, Electricity(ECL), Traffic and ILI.** These time series are relatively long, with the shortest dataset containing **more than 950** observations and **most ranging from 10,000 to 70,000 observations.** For a more manageable evaluation with LLMTime, we use smaller subsets from each time series, specifically selecting the last univariate series, "OT", and the final 96 and 192 time steps from each series for testing. The models are evaluated using the same model settings as in LLMTime, with additional comparisons against more advanced state-of-the-art TS forecasting models, including Informer , Autoformer, NSTransformer , TimesNet , PatchTST , and iTransformer.

Datasets	NLTS		iTransformer		LLMTime		TimesNet		NSTransformer		Autoformer		ARIMA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Synthetic TS (avg.)	**0.014**	**0.091**	1.215	0.878	0.016	0.102	1.193	0.874	1.197	0.871	0.029	0.135	1.234	0.886
Latest Stock (avg.)	**0.003**	**0.022**	0.184	0.229	0.030	0.056	0.189	0.207	0.131	0.194	0.183	0.308	0.007	0.041
Wireless Traffic(avg.)	**0.016**	**0.084**	1.499	0.832	0.017	0.087	1.546	0.899	1.146	0.712	1.385	0.786	0.026	0.111
Smart Glasses(avg.)	**0.001**	**0.022**	10.096	4.026	0.003	0.030	62.501	4.935	95.378	5.273	58.471	5.194	0.002	0.027

Table 6: Average zero-shot forecasting performance on synthetic and latest stock TS datasets. **Bold** and underline: the best and the second best results in each row.



(a) Synthetic TS (b) Latest Stock TS

(c) Wireless Traffic (d) Smart Glasses

Figure 6: Average zero-shot forecasting performance on the synthetic dataset and the latest stock market datasets.

E.2 Implementation

We utilize the GPyTorch library to implement Gaussian processes (GPs), and the Darts library for modeling with ARIMA, TCN, N-BEATS, and N-HiTS. For any hyperparameters not detailed below, we use the default settings. For all LLMs and non-LLM models, we consider the following model settings:

- GPT-3.5: $\alpha = 0.95$, temperature = 0.7, $\beta = 0.3$, basic = False, and precision = 3, with serializer settings: base = 10, signed = True, and half bin correction = True.
- Gemini, Claude, GLM, Qwen, Deepseek: $\alpha = 0.95$, temperature = 1.0, top P = 0.8, basic = True, precision = 3.
- Spectral Mixture Gaussian Process (SM-GP): We use a Gaussian Process with a kernel consisting of a spectral mixture kernel with 12 components and an RBF kernel. The learning rate is tuned from $\{5e-3, 1e-2, 5e-2, 1e-1\}$.
- ARIMA: We perform a grid search over $p \in \{12, 20, 30\}$, $d \in \{1, 2\}$, and $q \in \{0, 1, 2\}$.
- TCN: We perform a grid search over input_chunk_length $\in \{10, 100, 400\}$, output_chunk_length $\in \{1, 10\}$, kernel_size $\in \{3, 5\}$, num_filters $\in \{1, 3\}$, and likelihood $\in \{\text{Laplace}, \text{Gaussian}\}$.
- N-BEATS: We perform a grid search over input_chunk_length $\in \{10, 100, 400\}$, output_chunk_length $\in \{1, 10\}$, layer_widths $\in \{64, 16\}$, num_layers $\in \{1, 2\}$, and likelihood $\in \{\text{Laplace}, \text{Gaussian}\}$.
- N-HiTS: We perform a grid search over input_chunk_length $\in \{10, 100, 400\}$, output_chunk_length $\in \{1, 10\}$, layer_widths $\in \{64, 16\}$, num_layers $\in \{1, 2\}$, and likelihood $\in \{\text{Laplace}, \text{Gaussian}\}$.

Note that we do not use the latest or most advanced LLMs, especially higher versions of Chat-

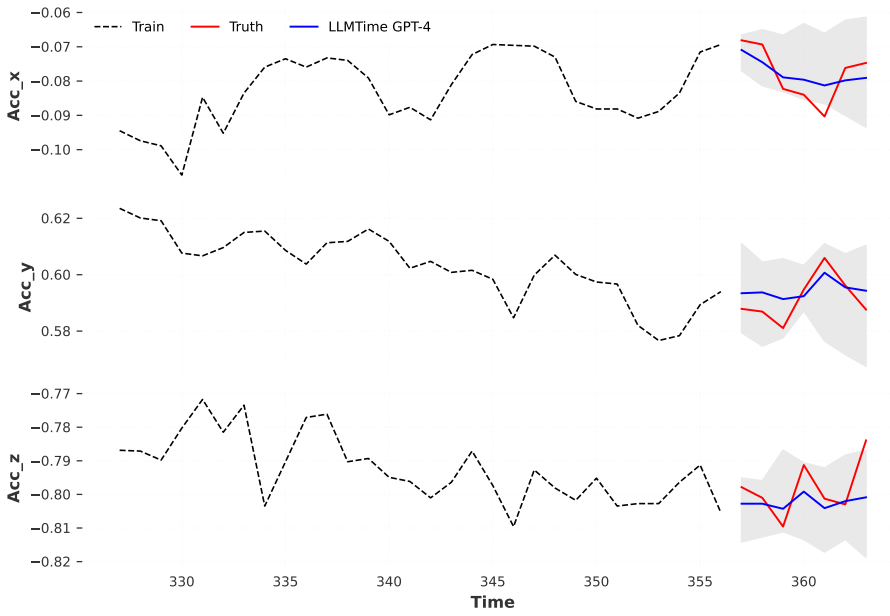


Figure 7: Forecasting of GPT-4 with NLTS on the UESTC-MMEA-CL dataset.

GPT, due to the cost issues. The expense of advanced ChatGPT models could be many times that of the standard version. On the other hand, the LLMs currently used are quite popular. These models have considerable representativeness and diversity. Furthermore, the focus of the study is to explore whether adding noise to TS can enhance the forecasting ability of LLMs. We believe that the findings obtained on existing LLMs can provide valuable references and insights for the optimization of more advanced LLMs in the future.

E.3 Access cost of different LLMs

In Table 7, we present the access cost of different LLMs used for both prompt and completion tasks. Each model is listed with the associated costs for processing 1,000 tokens in prompt and completion tasks. We provide a clear comparison of the token pricing for both types of tasks, ranging from as low as $ 0.00021 per 1 K tokens to as high as $ 0.16 per 1 K tokens. It is important to acknowledge that the table presents a single snapshot of historical pricing. Indeed, the pricing of LLMs is subject to flux, largely influenced by the competitive dynamics within the commercial landscape of LLM providers.

The cheapest models for prompts are the Qwen3-4B. The most expensive model for prompts and completion tokens is Claude-3.5-Opus, costing 0.032 and 0.16 for 1,000 tokens. When choosing a model, it's important to consider both the cost of prompts and completions. Some models are cheaper for prompts but not for completions, and vice versa. Practitioners should weigh the costs against the performance benefits to decide which model is best for their needs, especially when dealing with high volumes of data.

E.4 Performance of diverse LLMs.

Table 8 reports the performance of nine different LLMs, including GPT-4, GPT-3.5-Turbo-Instruct, Moonshot-V1-8k, Claude-3-Opus, Claude-3.5-Sonnet, Claude-3.5-Haiku, DeepSeek-V3, GLM-4-Air, and Qwen3-4B, under varying noise levels on the ETTh2 dataset. Due to space constraints, abbreviated model names are used in the tables. Across most noise levels, all evaluated LLMs exhibit a certain degree of performance improvement compared to the Original condition. Most LLMs exhibit more noticeable performance improvements under low-noise conditions. For example, Claude-3.5-Sonnet and GLM-4-Air achieve the lowest errors at noise levels such as 0.001–0.005 compared to the original setting. This indicates that injecting noise with an appropriate intensity can effectively enhance the predictive performance. The sensitivity to noise varies significantly across models. Some maintain stable performance as the noise level increases, while others experience noticeable fluctuations. This disparity may stem from our

LLMs	Number of prompt tokens	Prompt price	Number of completion tokens	Completion price
GPT-4	1 K	$ 0.03	1 K	$ 0.06
GPT-3.5-Turbo-Instruct	1 K	$ 0.0015	1 K	$ 0.002
Claude-3-Opus	1 K	$ 0.032	1 K	$ 0.16
Claude-3.5-Haiku	1 K	$ 0.00233	1 K	$ 0.01167
Claude-3.5-Sonnet	1 K	$ 0.007	1 K	$ 0.035
Deepseek-V3	1 K	$ 0.0005	1 K	$ 0.002
GLM-4-Air	1 K	$ 0.0005	1 K	$ 0.0005
Qwen3-4B	1 K	$ 0.00021	1 K	$ 0.00084

Table 7: Prices of LLMs for prompt and completion tasks.

Off-the-Shelf setting without any fine-tuning or additional pretraining, with the outcome largely influenced by the model's architecture, scale, and the diversity of its pretraining data.

E.5 Performances of different forecasting horizons

We comprehensively evaluate the performance of several forecasting models across two forecasting horizons, 96 and 192, following the experimental settings established by LLMTime, using seven widely adopted benchmark datasets. As shown in Table 9 and Table 10, our model consistently outperforms all baseline models at both horizons, demonstrating strong adaptability and generalization in capturing temporal features.

E.6 Comparative performance of noise levels

Our results are based on GPT-3.5-Turbo-Instruction. As shown in Table 11, injecting noise at various levels consistently improves forecasting performance across all datasets compared to the original input. However, sensitivity to noise levels varies among datasets, and there is no single noise level that universally optimizes performance. The effectiveness of noise injection is influenced by multiple factors, including data distribution, task complexity, and model capacity. More specifically, experiments demonstrate that moderate noise levels (approximately 0.005 to 0.02) yield the best performance improvements, as such noise helps the model capture key features in the data and enhances prediction accuracy. In contrast, higher noise levels (e.g., 0.05) may excessively disrupt the semantic structure of inputs. In future work, we will focus on identifying the optimal noise level and type tailored to different tasks.

E.7 Impact of sampling strategy

To investigate how sampling strategies affect the robustness of LLMTime, we evaluate its performance on the ETTh2 dataset by varying both the trial count (1, 5, 10, 15, 20) and the aggregation method (mean vs. median). As shown in Table 12 and Table 13, the aggregation strategy and trial count jointly impact performance. Specifically, moderately increasing the trial count from 1 to 5 or 10 generally leads to noticeable improvements in MSE and MAE, enhancing result stability and accuracy. However, further increases to 15 or 20 yield diminishing returns. Moreover, median aggregation consistently produces better optimal results than mean aggregation, indicating its effectiveness in mitigating outliers and reducing variability.

Balancing both model performance and computational cost, we adopt a trial count of 10 and median aggregation for our main experiments.

E.8 Forecasting performance under different noise types

Table 14 reports the MSE and MAE results of our method on the ETTh2 dataset. The evaluation is conducted under six different noise types: Uniform, Laplace, Geometric, Gamma, Beta, and Gaussian. The definitions of these noise distributions are provided in Appendix C.1.

E.9 Average forecasting performance of models

For short- and long-term forecasting tasks, we evaluate three major TS benchmarks: Darts, Memorization, and Autoformer, which contain 8, 3, and 7 sub-datasets. For each benchmark, we report the average performance across its sub-datasets. For Autoformer, following the LLMTime setting, we present results for forecasting horizons of 96 and 192 steps, as illustrated in Fig. 8

E.10 Visualization of forecasting across all datasets

Due to the limited page space, we selectively present the forecasting plots. Representative examples are provided in Fig. 9 - Fig. 13.

Noise Level	GPT-4		GPT-3.5		Moonshot		Claude-Opus		Claude-Sonnet		Claude-Haiku		Deepseek		GLM		Qwen	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	0.064	0.186	0.042	0.148	0.021	0.114	0.012	0.084	0.044	0.176	0.0169	0.095	0.0064	0.0648	0.024	0.128	0.273	0.355
0.001	0.033	0.144	0.043	0.146	0.015	0.099	0.011	0.080	0.007	**0.065**	0.0162	0.093	0.0064	0.0648	0.016	0.109	0.264	0.324
0.005	0.054	0.190	0.039	0.144	0.017	0.103	0.010	0.073	**0.006**	**0.065**	0.0148	0.091	0.0063	0.0636	**0.007**	**0.064**	0.260	0.311
0.01	0.047	0.166	0.039	0.138	**0.013**	**0.085**	0.008	0.072	0.035	0.151	0.0094	0.073	0.0064	0.0643	0.009	0.075	**0.258**	0.317
0.02	**0.028**	**0.132**	**0.024**	**0.117**	0.023	0.117	**0.007**	**0.061**	0.050	0.183	**0.0091**	**0.070**	**0.0061**	**0.0625**	0.008	0.071	0.265	0.331
0.05	0.060	0.180	0.037	0.146	0.015	0.103	0.008	0.064	0.033	0.147	0.0093	0.074	0.0063	0.0628	0.017	0.093	0.084	**0.251**

Table 8: MSE and MAE of LLMs on ETTh2 under varying noise levels. "Original" refers to results obtained using LLMTime without noise injection for the same LLM.

Model	ECL		ETTh1		ETTh2		ETTm1		ETTm2		Traffic		ILI	
	96	192	96	192	96	192	96	192	96	192	96	192	24	36
Informer	0.124	0.195	0.039	0.149	0.174	0.363	0.004	0.019	0.075	0.160	0.079	0.171	11.047	9.579
Autoformer	0.235	0.534	0.071	0.073	0.184	0.210	0.009	0.102	0.077	0.449	0.073	0.560	1.077	0.662
TimesNet	0.076	0.246	0.080	0.058	0.314	0.220	0.008	0.025	0.021	0.361	0.077	0.383	0.766	0.700
PatchTST	0.124	0.303	0.071	0.059	0.331	0.264	0.006	0.037	0.026	0.403	0.075	0.439	0.979	1.048
LLMTime	0.011	0.011	0.008	0.030	0.042	0.026	0.004	0.019	0.028	0.083	0.010	0.019	0.084	0.014
iTransformer	0.084	0.252	0.038	0.073	0.214	0.264	0.003	0.031	0.021	0.385	0.091	0.403	0.426	0.868
Ours	**0.010**	**0.009**	**0.005**	**0.016**	**0.024**	**0.018**	**0.002**	**0.009**	**0.009**	**0.048**	**0.001**	**0.007**	**0.016**	**0.007**

Table 9: MSE of different models across multiple datasets and forecasting horizons. **Bold**: the best result under each forecasting horizon, Underline: the second best

Model	ECL		ETTh1		ETTh2		ETTm1		ETTm2		Traffic		ILI	
	96	192	96	192	96	192	96	192	96	192	96	192	24	36
Informer	0.291	0.349	0.164	0.329	0.324	0.503	0.054	0.109	0.250	0.354	0.221	0.338	3.312	3.065
Autoformer	0.364	0.535	0.234	0.208	0.324	0.379	0.078	0.300	0.249	0.591	0.213	0.642	0.969	0.633
TimesNet	0.208	0.338	0.262	0.175	0.483	0.389	0.069	0.138	0.124	0.534	0.228	0.509	0.765	0.619
PatchTST	0.289	0.370	0.239	0.181	0.478	0.431	0.056	0.170	0.139	0.554	0.227	0.555	0.687	0.809
LLMTime	0.127	0.123	0.077	0.143	0.148	0.131	0.045	0.123	0.142	0.236	0.069	0.111	0.115	0.103
iTransformer	0.210	0.361	0.163	0.205	0.359	0.429	0.044	0.159	0.117	0.555	0.257	0.546	0.486	0.764
Ours	**0.076**	**0.078**	**0.052**	**0.097**	**0.117**	**0.112**	**0.035**	**0.077**	**0.077**	**0.185**	**0.027**	**0.070**	**0.106**	**0.071**

Table 10: MAE of different models across multiple datasets and forecasting horizons. **Bold**: the best result under each forecasting horizon, Underline: the second best

Noise Level	Traffic		ETTh2		AusBeer		GasRateCO2		MonthlyMilk	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	0.010	0.069	0.042	0.148	0.0011	0.026	0.038	0.160	0.0054	0.062
0.001	0.004	0.047	0.043	0.146	0.0009	0.024	0.034	0.150	0.0053	0.058
0.005	0.002	0.033	0.039	0.144	**0.0007**	**0.020**	0.025	0.124	**0.0024**	**0.042**
0.01	0.002	0.034	0.039	0.138	0.0008	0.021	**0.019**	**0.106**	0.0035	**0.042**
0.02	**0.001**	**0.027**	**0.024**	**0.117**	0.0008	0.022	0.022	0.121	0.0044	0.054
0.05	0.005	0.058	0.037	0.146	0.0011	0.025	0.035	0.151	0.0033	0.045

Table 11: MSE and MAE across different noise levels for multiple datasets. **Bold**: the best result for each dataset

Noise Level	Trial count = 1		Trial count = 5		Trial count = 10		Trial count = 15		Trial count = 20	
	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean
original	0.067	0.055	0.044	0.050	0.042	0.036	0.050	0.053	0.042	0.048
0.001	0.034	0.026	0.033	**0.017**	0.043	0.041	**0.015**	0.038	0.027	0.046
0.005	0.031	0.051	**0.030**	**0.027**	0.039	**0.031**	0.027	**0.029**	0.033	0.037
0.01	**0.019**	**0.025**	0.031	0.034	0.039	0.035	0.038	0.038	0.041	0.036
0.02	0.036	0.036	0.037	0.033	**0.024**	0.037	0.047	0.038	0.031	0.035
0.05	0.057	0.031	0.046	0.036	0.037	0.040	**0.035**	0.033	0.035	0.037

Table 12: MSE under different trial counts and aggregation methods. "Trial count" refers to the number of repeated runs. "Mean" denotes averaging the results across all runs, while "Median" denotes taking the median across time steps within each run.

Noise Level	Trial count = 1		Trial count = 5		Trial count = 10		Trial count = 15		Trial count = 20	
	Median	Mean	Median	Mean	Median	Mean	Median	Mean	Median	Mean
original	0.198	0.181	0.147	0.163	0.148	0.153	0.179	0.177	0.159	0.169
0.001	0.160	0.127	0.137	**0.103**	0.146	0.157	**0.093**	0.150	**0.129**	0.164
0.005	0.133	0.167	0.138	0.136	0.144	**0.135**	0.121	**0.130**	0.139	0.154
0.01	**0.111**	0.137	0.137	0.138	0.138	0.146	0.150	0.147	0.154	0.146
0.02	0.163	0.159	**0.130**	0.137	**0.117**	0.148	0.174	0.152	0.132	0.148
0.05	0.186	**0.124**	0.172	0.143	0.146	0.146	0.144	0.142	0.140	**0.144**

Table 13: MAE under different trial counts and aggregation methods.

Noise Level	Uniform		Laplace		Geometric		Gamma		Beta		Gaussian	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	0.046	0.173	0.036	0.147	0.044	0.160	0.038	0.155	0.046	0.172	0.042	0.148
0.001	0.031	0.146	0.027	0.133	0.024	0.123	**0.025**	**0.120**	**0.028**	**0.124**	0.043	0.146
0.005	0.034	0.149	**0.024**	**0.118**	0.040	0.154	0.034	0.144	0.036	0.139	0.039	0.144
0.01	0.036	0.154	0.032	0.133	0.043	0.152	0.030	0.137	0.040	0.154	0.039	0.138
0.02	**0.025**	**0.119**	0.035	0.136	**0.024**	**0.120**	0.031	0.139	0.039	0.139	**0.024**	**0.117**
0.05	0.030	0.134	0.027	0.131	0.034	0.141	0.028	0.121	0.034	0.137	0.037	0.146

Table 14: MSE and MAE under different noise types.**Bold**: the best result for each noise level.

Datasets	NLTS		LLMTime		N-HiTS		N-BEATS		TCN		SM-GP		ARIMA	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Darts (avg.)	**0.015**	**0.082**	0.025	0.110	0.044	0.154	0.052	0.165	0.037	0.152	0.043	0.159	0.024	0.117
Memorization (avg.)	**0.020**	**0.073**	0.046	0.126	0.092	0.178	0.149	0.287	0.043	0.130	0.086	0.199	0.207	0.314

Table 15: Average zero-shot forecasting performance on the short-term benchmarks. **Bold** and underline: the best and the second-best results.

Prediction Length	NLTS		iTransformer		LLMTime		PatchTST		TimesNet		Autoformer		Informer	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Monash (avg.)	**0.022**	**0.102**	0.274	0.376	0.036	0.129	0.309	0.391	0.255	0.357	0.473	0.522	0.912	0.770
Autoformer (avg.)	**0.021**	**0.078**	0.125	0.234	0.029	0.103	0.230	0.302	0.192	0.306	0.247	0.347	1.649	0.660

Table 16: Average zero-shot forecasting performance on the long-term benchmarks. **Bold** and underline: the best and the second best results.

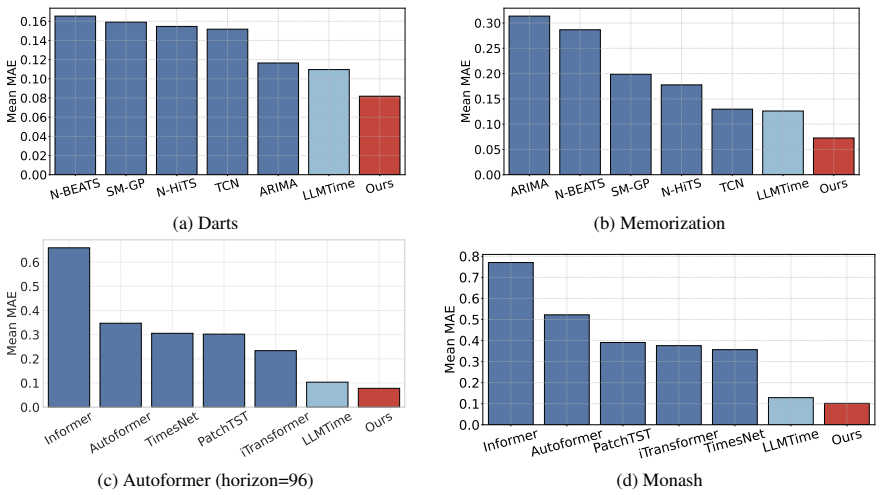
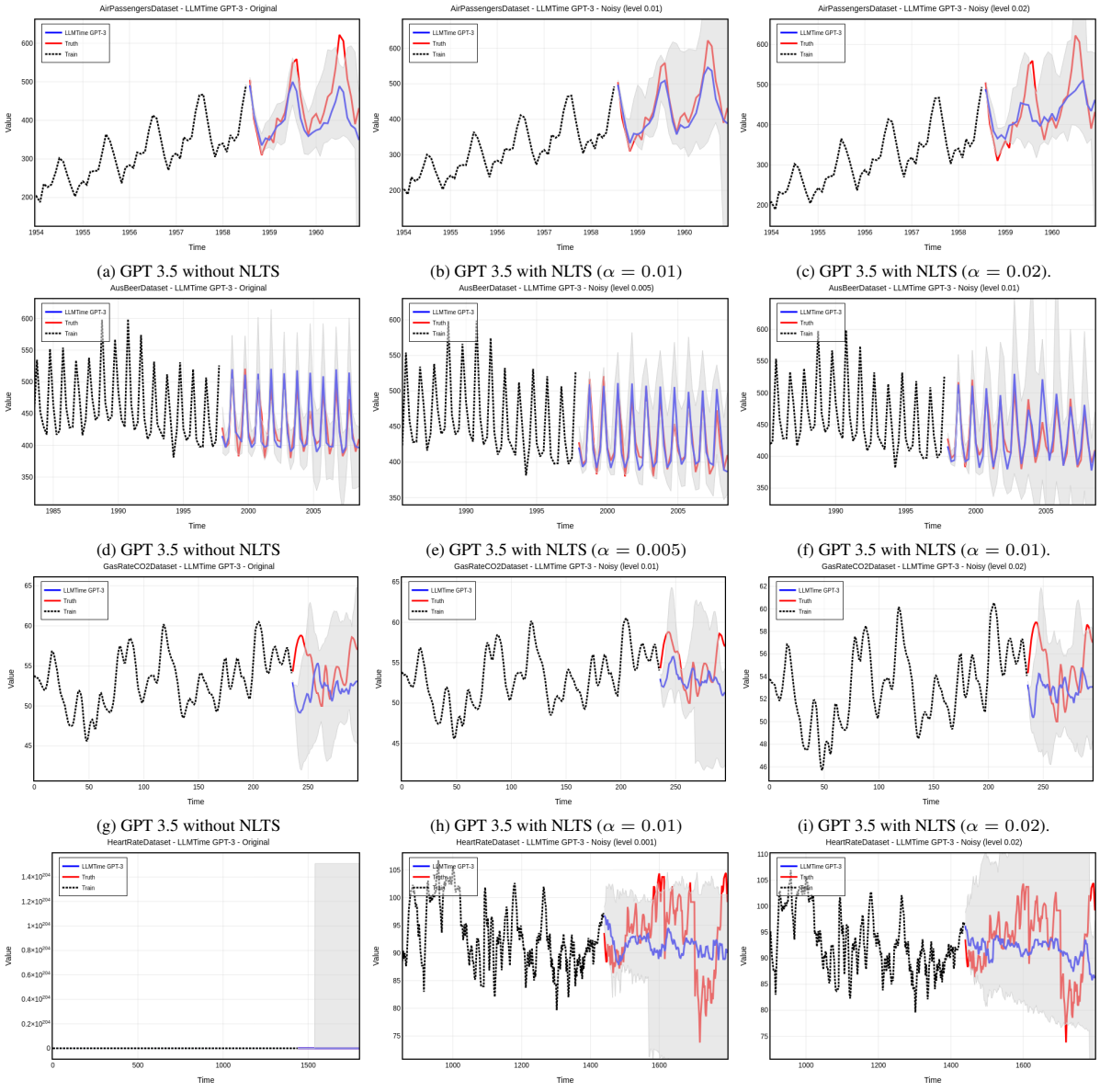


Figure 8: Average zero-shot forecasting performance (MAE) across the Darts, Memorization, Autoformer and Monash benchmarks.



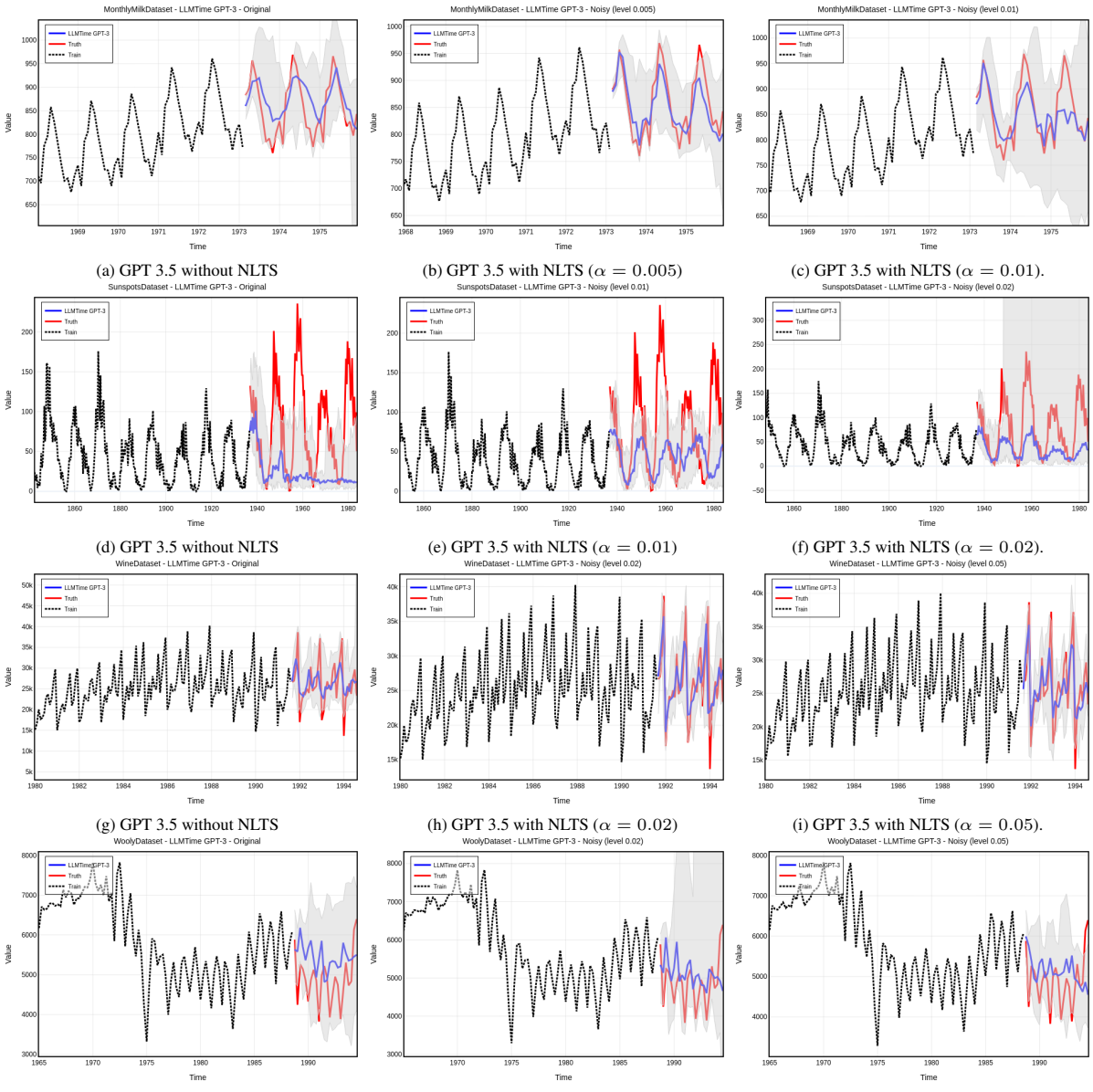
(a) GPT 3.5 without NLTS (b) GPT 3.5 with NLTS ($\alpha = 0.01$) (c) GPT 3.5 with NLTS ($\alpha = 0.02$).

(d) GPT 3.5 without NLTS (e) GPT 3.5 with NLTS ($\alpha = 0.005$) (f) GPT 3.5 with NLTS ($\alpha = 0.01$).

(g) GPT 3.5 without NLTS (h) GPT 3.5 with NLTS ($\alpha = 0.01$) (i) GPT 3.5 with NLTS ($\alpha = 0.02$).

(j) GPT 3.5 without NLTS (k) GPT 3.5 with NLTS ($\alpha = 0.001$) (l) GPT 3.5 with NLTS ($\alpha = 0.02$).

Figure 9: Forecasting results of GPT-3.5 on multiple time series from the Darts dataset under different noise injection levels. Each row corresponds to a specific dataset (AirPassengers, AusBeer, GasRateCO2, and HeartRate), with (a) showing the original prediction without noise, and (b)–(c) showing predictions with increasing levels of noise (α).



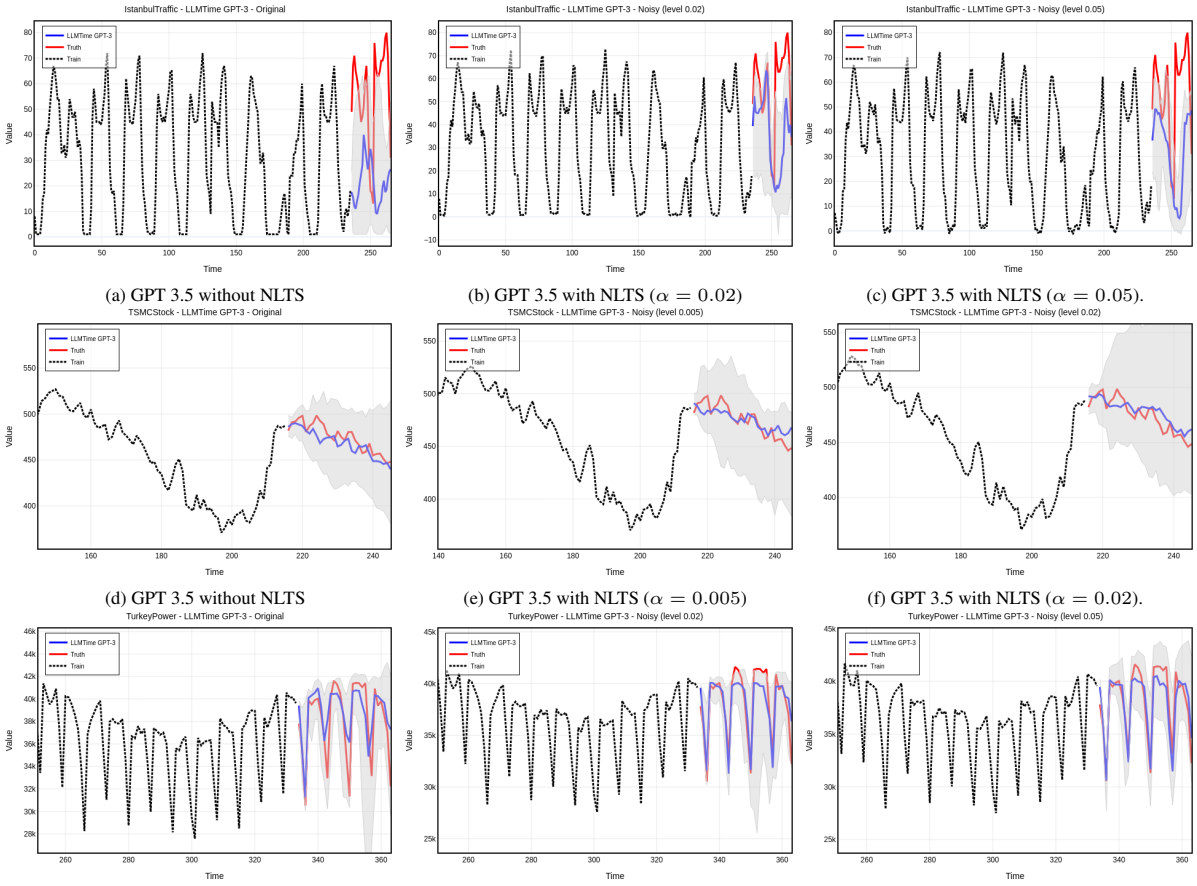
(a) GPT 3.5 without NLTS (b) GPT 3.5 with NLTS ($\alpha = 0.005$) (c) GPT 3.5 with NLTS ($\alpha = 0.01$).

(d) GPT 3.5 without NLTS (e) GPT 3.5 with NLTS ($\alpha = 0.01$) (f) GPT 3.5 with NLTS ($\alpha = 0.02$).

(g) GPT 3.5 without NLTS (h) GPT 3.5 with NLTS ($\alpha = 0.02$) (i) GPT 3.5 with NLTS ($\alpha = 0.05$).

(j) GPT 3.5 without NLTS (k) GPT 3.5 with NLTS ($\alpha = 0.02$) (l) GPT 3.5 with NLTS ($\alpha = 0.05$).

Figure 10: Forecasting results of GPT-3.5 on multiple time series from the Darts dataset under different noise injection levels. Each row corresponds to a specific dataset (MonthlyMilk, Sunspots, Wine, Woolly), with (a) showing the original prediction without noise, and (b)–(c) showing predictions with increasing levels of noise (α).

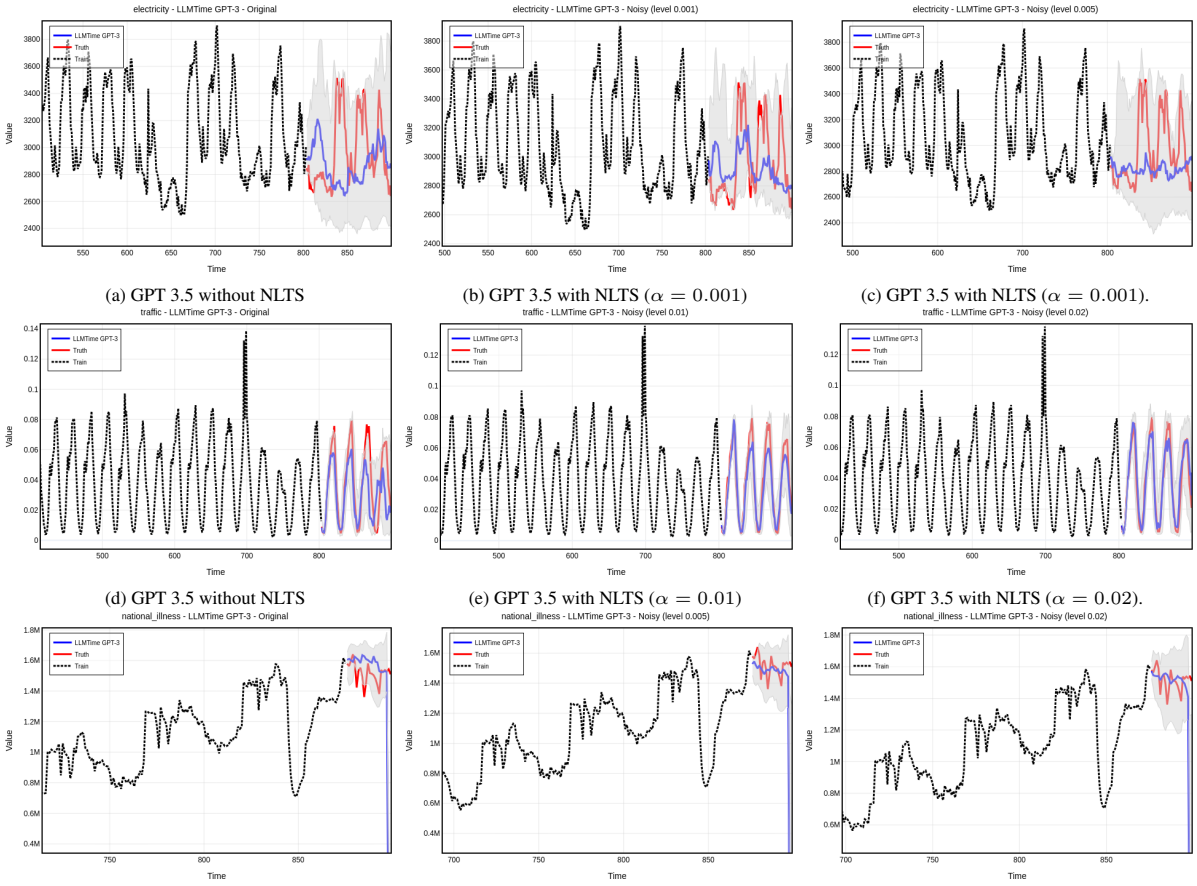


(a) GPT 3.5 without NLTS (b) GPT 3.5 with NLTS ($\alpha = 0.02$) (c) GPT 3.5 with NLTS ($\alpha = 0.05$).

(d) GPT 3.5 without NLTS (e) GPT 3.5 with NLTS ($\alpha = 0.005$) (f) GPT 3.5 with NLTS ($\alpha = 0.02$).

(g) GPT 3.5 without NLTS (h) GPT 3.5 with NLTS ($\alpha = 0.02$) (i) GPT 3.5 with NLTS ($\alpha = 0.05$).

Figure 11: Forecasting results of GPT-3.5 on multiple time series from the Memorization dataset under different noise injection levels. Each row corresponds to a specific dataset (IstanbulTraffic, TSMCStock, TurkeyPower), with (a) showing the original prediction without noise, and (b)–(c) showing predictions with increasing levels of noise (α).

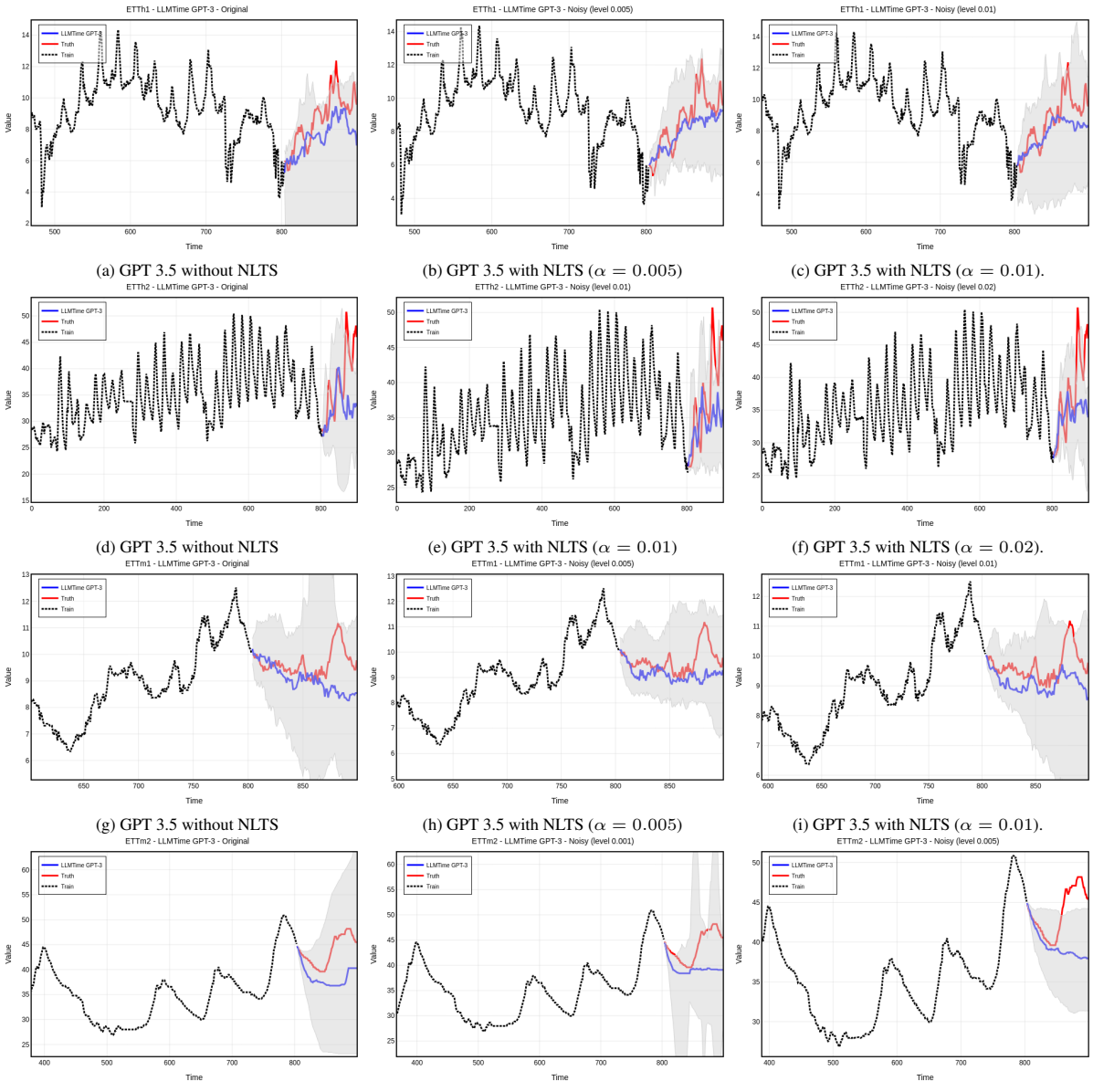


(a) GPT 3.5 without NLTS (b) GPT 3.5 with NLTS ($\alpha = 0.001$) (c) GPT 3.5 with NLTS ($\alpha = 0.001$).

(d) GPT 3.5 without NLTS (e) GPT 3.5 with NLTS ($\alpha = 0.01$) (f) GPT 3.5 with NLTS ($\alpha = 0.02$).

(g) GPT 3.5 without NLTS (h) GPT 3.5 with NLTS ($\alpha = 0.005$) (i) GPT 3.5 with NLTS ($\alpha = 0.02$).

Figure 12: Forecasting results of GPT-3.5 on multiple time series from the Autoformer dataset under different noise injection levels. Each row corresponds to a specific dataset (Electricity, Traffic, ILI), with (a) showing the original prediction without noise, and (b)–(c) showing predictions with increasing levels of noise (α).



(a) GPT 3.5 without NLTS (b) GPT 3.5 with NLTS ($\alpha = 0.005$) (c) GPT 3.5 with NLTS ($\alpha = 0.01$).

(d) GPT 3.5 without NLTS (e) GPT 3.5 with NLTS ($\alpha = 0.01$) (f) GPT 3.5 with NLTS ($\alpha = 0.02$).

(g) GPT 3.5 without NLTS (h) GPT 3.5 with NLTS ($\alpha = 0.005$) (i) GPT 3.5 with NLTS ($\alpha = 0.01$).

(j) GPT 3.5 without NLTS (k) GPT 3.5 with NLTS ($\alpha = 0.001$) (l) GPT 3.5 with NLTS ($\alpha = 0.005$).

Figure 13: Forecasting results of GPT-3.5 on multiple time series from the Autoformer dataset under different noise injection levels. Each row corresponds to a specific dataset (ETTh1, ETTh2, ETTm1, ETTm2), with (a) showing the original prediction without noise, and (b)–(c) showing predictions with increasing levels of noise (α).