Apollo-Forecast: Overcoming Aliasing and Inference Speed Challenges in Language Models for Time Series Forecasting

Tianyi Yin[1,2], Jingwei Wang[2,3*], Yunlong Ma[2*], Han Wang[2], Chenze Wang[2], Yukai Zhao[2], Min Liu[2], Weiming Shen[4]

[1]Shanghai Research Institute for Intelligent Autonomous Systems, Tongji University
[2]College of Electronics and Information Engineering, Tongji University
[3]Ant Group
[4]Huazhong University of Science and Technology
{yin_tianyi, jwwang, evanma, 2210126, wangchenze, zhaoyukaijake, lmin}@tongji.edu.cn,
wshen@ieee.org

Abstract

Encoding time series into tokens and using language models for processing has been shown to substantially augment the models' ability to generalize to unseen tasks. However, existing language models for time series forecasting encounter several obstacles, including aliasing distortion and prolonged inference times, primarily due to the limitations of quantization processes and the computational demands of large models. This paper introduces Apollo-Forecast, a novel framework that tackles these challenges with two key innovations: the Anti-Aliasing Quantization Module (AAQM) and the Race Decoding (RD) technique. AAQM adeptly encodes sequences into tokens while mitigating high-frequency noise in the original signals, thus enhancing both signal fidelity and overall quantization efficiency. RD employs a draft model to enable parallel processing and results integration, which markedly accelerates the inference speed for long-term predictions, particularly in large-scale models. Extensive experiments on various real-world datasets show that Apollo-Forecast outperforms state-of-the-art methods by 35.41% and 18.99% in WQL and MASE metrics, respectively, in zero-shot scenarios. Furthermore, our method achieves an acceleration of 1.9X-2.7X in inference speed over the baseline methods.

Code — https://github.com/Ivan-YinTY/Apollo-Forecast
Extended version — https://arxiv.org/abs/2412.12226

Introduction

Time series forecasting plays a pivotal role in various research domains, including transportation (Afrin and Yodo 2022), energy (Cai et al. 2024), and manufacturing (Wang et al. 2022). Early forecasting techniques predominantly relied on statistical methods such as ARIMA and machine learning models (Li, Wu, and Liu 2023). These approaches are often suitable for scenarios with limited observational data, offering a balanced performance. The emergence of deep learning techniques, such as NHITS (Challu et al. 2023) and Dlinear (Zeng et al. 2023), coinciding with the expansion of diverse time series data sources, has significantly enhanced forecasting capabilities (Tang et al. 2022). Nonetheless, these methods typically operate under a one-model-per-dataset framework and have yet to achieve a universally applicable forecasting model (Zhang et al. 2024).

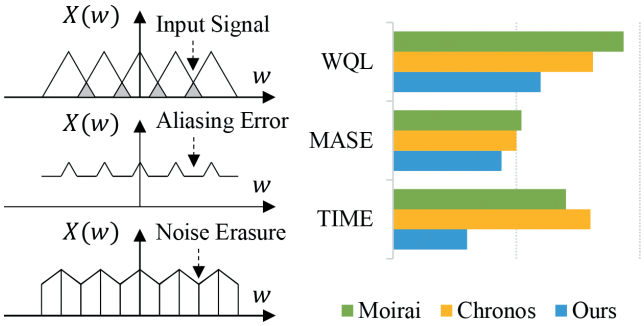

Figure 1: **(Left)** Without a noise erasure mechanism, quantization causes aliasing distortion. With a noise erasure mechanism, high-frequency noise is removed, preserving low-frequency information. X represents the frequency domain. **(Right)** Performance comparison of Moirai-L, Chronos-S, and Apollo-S on UCR dataset.

The emergence of large language models (LLMs) with robust generalization capabilities has inspired the development of foundation models for time-series forecasting. Those studies can be categorized into two main fields: enhancing pre-trained LLMs with prompts and retraining LLMs for time series tasks (Zhang et al. 2024). These methods bridge the gap between specialized models and general approaches, eliminating the need for retraining from scratch for each task (Jin et al. 2024a). Notably, the latter approach avoids errors introduced by the generic LLM vocabulary by encoding the time series into tokens, also known as quantization.

Despite the advantages of these foundation models, existing quantization methods may introduce high-frequency noise into the low-frequency band during tokenization, leading to additional distortion known as aliasing errors (Ma

*Corresponding authors: Yunlong Ma, Jingwei Wang.

et al. 2024), as illustrated in Figure 1. Furthermore, due to the autoregressive mechanism and the large model size, the response time of these models is relatively slow, especially in long-horizon scenarios (Cleaveland et al. 2024). These issues limit the model's performance in real situations and increase the cost of utilization.

In this paper, we propose a novel Apollo-Forecast [1] framework to address these challenges, incorporating two critical improvements: the Anti-Aliasing Quantization Module (AAQM) and Race Decoding (RD). AAQM introduces a Noise Erasure mechanism to eliminate interference components from the original signal, preventing information loss due to quantization with a fixed token length. The RD technique accelerates inference by employing a draft model (a faster model with reasonable accuracy) and a main model for parallel processing, directly merging the draft results with the main results after passing a Tolerance Check. We conducted extensive experiments across various datasets in multiple domains, demonstrating the generalization and speed enhancements of the proposed method.

Our contributions are summarized as follows:

1. We present the Anti-Aliasing Quantization module that substantially reduces aliasing errors caused by the encoding process.
2. We propose a novel decoding mechanism called Race Decoding to accelerate the inference speed of the foundation model without altering any structure.
3. We demonstrate through extensive experiments that our method improves performance by 35.41% in weighted quantization loss (WQL) and speed by 1.9X-2.7X compared to SOTA methods.

Related Work

Time Series Forecasting

Time series forecasting involves predicting future trends based on historical data and is widely used in transportation (Deng et al. 2022), manufacturing (He et al. 2023), energy (Zhang et al. 2023), and other fields. The methods for time series forecasting can be broadly categorized into three types: statistical models, machine learning models, and deep learning models (Chen et al. 2023).

Statistical Models Early approaches primarily utilized approaches such as ARIMA and EMA Model (Ansari et al. 2024). These models rely on mathematical formulations to capture underlying patterns in the data. ARIMA models are particularly effective for univariate time series and can handle non-stationary data through differencing. EMA Model, based on exponential smoothing, capturing seasonality and trend components (Du et al. 2022). While these models can produce reasonable outcomes from limited observations, they often require manual tuning and domain expertise.

Machine Learning Predictors Machine learning introduced more sophisticated models like Support Vector Machines (Vukovic et al. 2022), Random Forests, and Gradient Boosting Machines for time series forecasting (Li and Jung 2023). These models can capture complex patterns and interactions in the data without explicit mathematical formulations. Machine learning models often involve feature engineering to extract relevant features from raw time series data, enhancing predictive performance. However, they may struggle with capturing long-term dependencies and require careful hyperparameter tuning.

Deep Learning Predictors The development of deep learning has led to new feature extractors such as RNN-based, MLP-based, CNN-based, and Transformer-based models. RNNs, including LSTM and GRU, are particularly effective for sequential data as they capture temporal dependencies (Li and Jung 2023). CNNs can extract local patterns and features from time series data (He, Fu, and Lee 2022). Transformer-based models, which have shown impressive performance in natural language processing, are being adapted for time series forecasting due to their ability to capture long-range dependencies and parallelize computations (Chowdhury et al. 2022). Despite their outstanding performance, these deep learning methods are still limited to a one-model-per-dataset paradigm, making it challenging to extend a single model to datasets from different domains and failing to provide an integrated and general predictor.

LLMs for Time Series

Encoding time series as digits and processing them with LLMs offers a general approach without the need to train from scratch for different datasets (Nate Gruver and Wilson 2023). Current research bridges the gap between different modalities through three methods: prompts, multimodal alignment, and discretization. The prompt-based method uses natural language to describe the data, enabling the foundation model to develop analytical capabilities through in-context learning or fine-tuning (Jin et al. 2024b). Multimodal alignment maps the feature representations of time series to the text space through contrastive learning, allowing time series data to be directly input into LLMs for prediction. Discretization converts observed data into special discrete identifiers (IDs) and then passes them to the foundation model, avoiding the shortcomings of traditional dictionaries in distinguishing numbers. However, these methods are limited by token length requirements during the conversion process, leading to aliasing distortion that affects encoding accuracy. Additionally, these models are often based on LLMs such as LLaMA and GPT-2 (Rasul et al. 2024; Jin et al. 2023), which have slower inference speeds, impacting practical use. This study aims to address these issues.

Methodology

The architecture of the Apollo-Forecast framework is illustrated in Figure 2. It consists of three modules: the Anti-Aliasing Quantization Module (AAQM), the time series forecasting model (TSFM), and the Race Decoding (RD). Each component plays a distinct role within the framework, and their details will be introduced in the following.

[1] Apollo is recognized as the god of truth and prophecy.

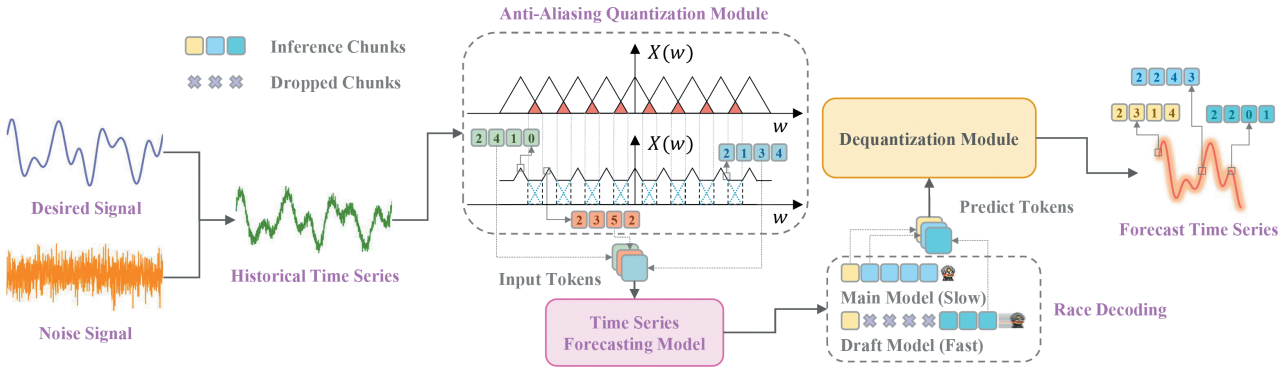

Figure 2: The architecture of Apollo-Forecast. The time series first passes through the **AAQM** to be converted into tokens, and then the forecasting model with **Race Decoding** is used to predict the next value. X represents the frequency domain.

Anti-Aliasing Quantization Module

When the token sequence length is fixed and cannot meet the requirements of the Nyquist sampling theorem, high-frequency noise will fold back into the low-frequency band, affecting the signal quality (Ma et al. 2024). To address this issue, we propose a novel method for time series tokenization called AAQM. This module comprises a noise erasure and a time series embedding module to achieve higher quality temporal signal encoding, as shown in Algorithm 1.

Noise Erasure Specifically, in the noise erasure (NE) mechanism, we employ a Butterworth filter to remove high-frequency noise from the raw signal. Let $x[n]$ be the original discrete-time signal, and $X[k]$ its Discrete Fourier Transform (DFT). The filter is designed to have a maximally flat frequency response in the passband. The transfer function $H(z)$ of an n-th order Butterworth filter in the z-domain is given by:

$$H(z) = \frac{1}{\sqrt{1 + \left(\frac{z}{\omega_c}\right)^{2n}}}, \tag{1}$$

where ω_c is the low cutoff frequency, and $z = e^{j\omega}$ is the complex frequency variable. The filtered signal $y[n]$ is obtained by applying the inverse DFT ($\mathcal{F}^{-1}$) to the product of $X[k]$ and $H(e^{j\omega})$:

$$y[n] = \mathcal{F}^{-1}\left\{X[k] \cdot H(e^{j\omega})\right\}. \tag{2}$$

Assume the input signal $x[n]$ is composed of the desired signal $s[n]$ and high-frequency noise $e[n]$, i.e., $x[n] = s[n] + e[n]$. The power spectral densities of signal and noise are $S[k]$ and $S_e[k]$, respectively. Before applying the NE mechanism, the signal-to-noise ratio (SNR) can be expressed as:

$$\text{SNR} = \frac{\sum_{k=0}^{N-1} |S[k]|^2}{\sum_{k=0}^{N-1} S_e[k]}, \tag{3}$$

After applying the NE operation, the power of the filtered signal P_s' and noise P_n' are:

$$P_s' = \sum_{k=0}^{\omega_c} |S[k]|^2, \tag{4}$$

$$P_n' = \sum_{k=0}^{\omega_c} S_e[k]|H(e^{j\omega})|^2, \tag{5}$$

Thus, the signal-to-noise ratio after filtering is:

$$\text{SNR}' = \frac{P_s'}{P_n'} = \frac{\sum_{k=0}^{\omega_c} |S[k]|^2}{\sum_{k=0}^{\omega_c} S_e[k]|H(e^{j\omega})|^2}, \tag{6}$$

Most real signals in nature usually have the following properties:

- The desired signal $s[n]$ has most of its energy concentrated in the low-frequency range $[0, \omega_c]$.
- The high-frequency noise $e[n]$ has significant energy in the high-frequency range $[\omega_c, N-1]$.

Based on these conditions, the NE mechanism effectively attenuates the high-frequency noise while preserving the low-frequency components of the input. Therefore, the power of the desired signal P_s' remains largely unchanged, while the power of the noise P_n' is significantly reduced.

Mathematically, this can be expressed as:

$$\sum_{k=0}^{\omega_c} |S[k]|^2 \approx \sum_{k=0}^{N-1} |S[k]|^2, \tag{7}$$

$$\sum_{k=0}^{\omega_c} S_e[k]|H(e^{j\omega})|^2 \ll \sum_{k=0}^{N-1} S_e[k], \tag{8}$$

Thus, we have:

$$\frac{\sum_{k=0}^{\omega_c} |S[k]|^2}{\sum_{k=0}^{\omega_c} S_e[k]|H(e^{j\omega})|^2} > \frac{\sum_{k=0}^{N-1} |S[k]|^2}{\sum_{k=0}^{N-1} S_e[k]}. \tag{9}$$

The proof demonstrates that the proposed method effectively reduces high-frequency noise, increasing the signal-to-noise ratio and reducing aliasing distortion. Compared to other filtering methods, our approach does not affect the phase, eliminating the need for phase compensation and thereby reducing computational load.

Algorithm 1: Anti-Aliasing Quantization

Input: $x[n]$, ω_c, f_s, n, Q
Output: Token sequence $\mathcal{T}$

$f_N \leftarrow \frac{f_s}{2}$
$\omega_N \leftarrow \frac{\omega_c}{f_N}$
Compute filter coefficients $b, a \leftarrow \text{butter}(n, \omega_N, \text{low})$
Apply zero-phase filtering $y[n] \leftarrow \text{filtfilt}(b, a, x[n])$
Normalize $y_{norm}[n] \leftarrow \frac{y[n]-\min(y[n])}{\max(y[n])-\min(y[n])}$
Quantize $y_q[n] \leftarrow \lfloor y_{norm}[n] \times Q \rfloor / Q$
Tokenize $\mathcal{T} \leftarrow \{t_i \mid t_i = \mathcal{E}(y_q[i]), \forall i \in [1, N]\}$
return $\mathcal{T}$

Time Series Embedding Module In order to use LLM for prediction, we convert the obtained high-quality series signal into a discrete token representation. In this process, the complete time series signal can be considered as $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, where N is the length of the sequence. The signal is first normalized and quantized as follows:

$$x_{norm} = \frac{\mathbf{x} - \min(\mathbf{x})}{\max(\mathbf{x}) - \min(\mathbf{x})}, \tag{10}$$

$$x_q = \lfloor x_{norm} \times Q \rfloor / Q, \tag{11}$$

where $Q = 10^4$ is the quantization factor, and $\lfloor \cdot \rfloor$ denotes the floor function, which rounds down to the nearest integer. The resulting quantized sequence is $\mathbf{x}_q = \{x_{q,1}, x_{q,2}, \ldots, x_{q,N}\}$. The quantized values are retained to four decimal places, ranging from 0.0000 to 1.0000, with a theoretical quantization error within the interval $[-\frac{1}{2Q}, \frac{1}{2Q}]$.

The tokenization process involves mapping the normalized and quantized signal $\mathbf{x}_q$ to a discrete set of tokens. The set of tokens is denoted as $\mathcal{T}$, and the embedding module as $\mathcal{E}$. The tokenization can be expressed as:

$$\mathcal{T}(\mathbf{x}_q) = \{t_i \mid t_i = \mathcal{E}(x_{q,i}), \forall i \in [1, N]\}, \tag{12}$$

Where N is the length of the time series. The embedding function $\mathcal{E}$ maps each quantized value to a token in the vocabulary. The resulting token sequence $\mathcal{T}(\mathbf{x}_q)$ is then used as input for the LLM.

Time Series Forecasting Model

By converting time series data into token sequences, we can predict the next time frame similarly to the seq2seq tasks in natural language processing. In this process, the TSFM can be viewed as a transformer that converts historical observations into forecast sequences, as illustrated by the following formula:

$$\hat{y}_{i+h} = f(t_i, t_{i-1}, \ldots, t_{i-n}; \theta), \tag{13}$$

Here, $\hat{y}_{i+h}$ represents the predicted value at time $i + h$, where h is the prediction horizon. The function f takes as input the historical token sequences $t_i, t_{i-1}, \ldots, t_{i-n}$, which are the observed values at times $i, i-1, \ldots, i-n$ respectively. The parameter θ denotes the model parameters.

Our approach involves predicting frames with a fixed horizon based on the input historical data and comparing them with the ground truth. The training loss is measured using the cross-entropy function:

$$\mathcal{L} = -\sum_{i=1}^{N} y_i \log(\hat{y}_i), \tag{14}$$

The data for training LLM includes various domains with different time granularities, such as finance, electricity, and transportation. This diversity enhances the model's generalization capability in zero-shot scenarios (Das et al. 2023).

Race Decoding

Despite the seq2seq model performing well, its inference speed significantly slows down when dealing with long-horizon prediction scenarios. Existing speculative decoding methods (Leviathan, Kalman, and Matias 2023) can accelerate large language model predictions at the token level, but they result in substantial computational waste for time series prediction tasks. Therefore, we propose the RD mechanism to accelerate the inference speed of TSFM at the chunk level while maintaining accuracy, as shown in Algorithm 2.

Draft Inference Models with fewer parameters generally have faster inference speeds under the same computational precision. In our method, when using a large model as the primary predictor for time series, a smaller, auxiliary model is automatically assigned as the draft predictor. Both predictors iteratively forecast the time series frames in parallel. Given the prediction horizon H, the inference time T for the primary and draft predictors can be defined as functions of H. As $H \to \infty$, the speed advantage of the draft predictor becomes more apparent, i.e., $T_{\text{main}}/T_{\text{draft}} \gg 1$. This implies that the draft predictor allows for quicker complete results as the horizon increases.

Ideally, during training, both the draft and primary predictors learn to approximate the same probability distribution $P(y|x)$ with different precisions. Let $\mathcal{D}$ represent the data distribution, and $\mathcal{L}$ be the loss function. The training objective for both predictors can be expressed as:

$$\theta^*_{\text{main}} = \arg\min_{\theta_{\text{main}}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(P_{\text{main}}(y|x;\theta_{\text{main}}), y)], \tag{15}$$

$$\theta^*_{\text{draft}} = \arg\min_{\theta_{\text{draft}}} \mathbb{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(P_{\text{draft}}(y|x;\theta_{\text{draft}}), y)]. \tag{16}$$

Here, θ_{main} and θ_{draft} represent the parameters of the primary and draft predictors, respectively. The notation θ^*_{main} and θ^*_{draft} denote the optimal parameters that minimize the expected loss $\mathcal{L}$ over the data distribution $\mathcal{D}$. With sufficient training, both predictors should ideally approximate the true distribution $P(y|x)$, with the errors due to model complexity denoted by ϵ_{main} and ϵ_{draft}:

$$P_{\text{main}}(y|x) \approx P(y|x) + \epsilon_{\text{main}}, \tag{17}$$

$$P_{\text{draft}}(y|x) \approx P(y|x) + \epsilon_{\text{draft}}. \tag{18}$$

When both predictors have sufficient representational capability to fit the current data distribution, the errors due to

Algorithm 2: Race Decoding Algorithm

Input: x, H
Output: y

Initialize θ_{main} and θ_{draft}
$t = 0$
while $t < H$ **do**
  Draft Inference:
  $P_{\text{draft}}(y|x; \theta_{\text{draft}})$
  $P_{\text{main}}(y|x; \theta_{\text{main}})$
  Tolerance Check:
  $\Delta P_t = \|P_{\text{main}}(y_{t_1:t_k}|x) - P_{\text{draft}}(y_{t_1:t_k}|x)\|$
  if $\Delta P_t < \gamma$ **then**
    Result Concatenation:
    Concatenate $P_{\text{main}}(y_{t_1:t_k}|x)$ with $P_{\text{draft}}(y_{t_{k+1}:t_H}|x)$
    return concatenated result
  else
    Wait for θ_{main} to complete inference
    return $P_{\text{main}}(y|x; \theta_{\text{main}})$
  end if
  $t = t + 1$
end while

model complexity ϵ_{main} and ϵ_{draft} also tend to zero. Thus, we can approximate:

$$\lim_{\epsilon_{\text{main}}, \epsilon_{\text{draft}} \to 0} P_{\text{main}}(y|x) \approx P_{\text{draft}}(y|x). \tag{19}$$

However, due to model randomness and distribution drift, the actual output distributions of the two predictors may differ. Therefore, after the draft predictor completes inference, an additional tolerance check step is required.

Tolerance Check Since the output probability distribution of the same TSFM under fixed temperature parameters is not prone to sudden changes, we can compare the inference results of the primary predictor with the corresponding frames of the draft predictor. When the time difference between the inferences of the two predictors is small, we can use the distribution P composed of partial frames as a proxy for the full sequence. Let $t_1 : t_k$ denote the frames predicted by the primary predictor and $t_{k+1} : t_H$ denote the frames predicted by the draft predictor. Then:

$$\Delta P_t = \|P_{\text{main}}(y_{t_1:t_k}|x) - P_{\text{draft}}(y_{t_1:t_k}|x)\|, \tag{20}$$

As ΔP_t approaches zero, the real difference between the main and draft predictors' output distributions can be expressed as:

$$\lim_{\Delta P_t \to 0} \|P_{\text{main}}(y|x) - P_{\text{draft}}(y|x)\| \approx 0, \tag{21}$$

When the two proxies are consistent in value, it can be approximated that the two outputs follow the same distribution. However, in time series prediction, the result fluctuation range is large, and strictly limiting consistency may render the acceleration mechanism ineffective. Therefore, we introduce an error factor γ to save computational cost while enhancing generalization, as shown in the following:

$$\|P_{\text{main}}(y_t|x) - P_{\text{draft}}(y_t|x)\| < \gamma. \tag{22}$$

This mechanism also achieves an ensemble learning effect, reducing the possibility of overfitting in large models.

Result Concatenation After passing the tolerance check, our method concatenates the existing inference results of the primary predictor with the remaining results of the draft predictor as the final result to further reduce errors. To compare the errors, let δ_{draft} be the error when using only the draft predictor, and η_{concat} be the error when concatenating the primary predictor's results with the draft predictor's results, as shown in the following equation:

$$\eta_{\text{concat}} = \sum_{t=1}^{k} \delta_{\text{main},t} + \sum_{t=k+1}^{H} \delta_{\text{draft},t}. \tag{23}$$

Assume that the primary predictor completes k frames, and the draft predictor completes the remaining $H - k$ frames. Since the error for each frame predicted by the primary predictor ($\delta_{\text{main},t}$) is typically smaller than the error for each frame predicted by the draft predictor ($\delta_{\text{draft},t}$), the total error η_{concat} when concatenating the results is smaller than the total error δ_{draft} when using only the draft predictor.

Given that the error for each frame predicted by the draft predictor is greater than or equal to the error for each frame predicted by the primary predictor, i.e., $\delta_{\text{draft},t} \geq \delta_{\text{main},t}$, it follows that the total error η_{concat} when concatenating the results is indeed smaller than the total error δ_{draft} when using only the draft predictor.

If the draft results do not pass the tolerance check, we continue to wait for the primary predictor to complete the inference and use it as the complete return result. The theoretical shortest time for the RD mechanism is the sum of the draft predictor inference time and the tolerance comparison time (extremely fast), and the longest time is the primary predictor inference time, mathematically expressed as:

$$T_{\text{RD}} = \min(T_{\text{draft}} + T_{\text{tolerance}}, T_{\text{main}}). \tag{24}$$

This ensures that the Race Decoding mechanism effectively accelerates the inference process while maintaining the accuracy of the predictions.

Experiments

Baselines and Experimental Settings

To evaluate the performance of the proposed method on real datasets, we selected over ten SOTA algorithms covering LLM-based (Chronos, Moirai), RNN-based (LSTM), MLP-based (N-HiTS, N-BEATS, DLinear), Transformer-based (TFT), and statistical models (AutoARIMA, AutoETS) as baselines for comparison experiments. Regarding datasets, we conducted extensive experiments on different datasets to verify the generalization of the proposed method under zero-shot conditions. In the experiments, we used five pre-trained Chronos-T5 models (Ansari et al. 2024) of different sizes as TSFM, namely Tiny (T), Mini (M), Small (S), Base (B), and Large (L), with a default horizon of 64. Meanwhile, the tiny model with AAQM is used as a draft model with a precision of float32, labeled Apollo-T*.

Method	Horizon = 448			Horizon = 512			Horizon = 576			Horizon = 640		
Metric	WQL	MASE	TIME	WQL	MASE	TIME	WQL	MASE	TIME	WQL	MASE	TIME
N-HiTS	8.027	17.942	2.151	7.945	17.914	2.578	5.219	18.542	2.771	8.236	19.212	3.116
N-BEATS	8.135	18.050	3.205	8.052	18.022	3.840	5.289	18.653	4.127	8.346	19.327	4.642
DLinear	9.043	19.325	**0.235**	8.950	19.295	**0.282**	5.879	19.971	**0.303**	9.278	20.693	**0.341**
TFT	7.394	18.309	1.608	7.319	18.281	1.927	4.807	18.921	2.071	7.586	19.605	2.329
Chronos-L	7.395	17.505	47.376	7.278	17.340	55.502	4.385	17.176	59.502	6.953	16.852	66.770
Chronos-B	8.162	19.296	25.591	7.882	19.479	29.868	5.090	19.591	31.777	7.782	18.874	35.693
Chronos-S	7.133	17.664	15.109	7.132	18.036	17.459	4.497	18.556	18.451	8.610	21.296	20.645
Chronos-M	9.264	18.967	12.240	9.344	18.460	15.392	6.842	20.611	16.842	9.470	21.708	19.039
Moirai-1.0-L	8.183	18.893	17.118	8.099	18.864	20.514	5.320	19.525	22.047	8.395	20.230	24.795
Moirai-1.0-B	8.350	19.649	10.285	8.264	19.619	12.326	5.429	20.306	13.247	8.567	21.040	14.897
AutoETS	10.010	20.600	23.429	9.908	20.569	28.076	6.508	21.289	30.175	10.270	22.059	33.935
AutoARIMA	9.091	19.628	261.739	8.997	19.598	313.664	5.910	20.284	337.105	9.327	21.017	379.111
Apollo-L	**6.583**	**17.447**	18.932	**6.137**	**17.219**	21.067	**4.300**	**17.166**	21.743	**6.338**	**16.849**	28.252
Apollo-B	7.223	17.465	13.923	6.779	17.719	18.464	4.382	17.805	20.117	6.904	16.901	20.699
Apollo-S	7.233	17.503	12.841	7.238	17.838	16.505	4.394	17.911	17.727	6.969	19.659	19.943
Apollo-M	7.350	17.580	12.111	7.469	17.999	15.365	4.419	18.304	16.744	7.231	19.957	18.821
Apollo-T*	6.245	17.207	11.819	5.854	17.128	14.651	4.250	17.166	16.387	6.197	16.809	18.292

Table 1: The performance of our Apollo-Forecast approach and other baseline models on the UCR dataset. WQL and MASE refer to Agg. Relative WQL and Agg. Relative MASE, respectively, while TIME refers to Avg. Inference Time (minutes). Apollo-T* represents the draft model.

Main Results

UCR Dataset In this experiment, we selected 114 available records from the UCR dataset for the experiment (Dau et al. 2019), covering multiple fields such as health, energy, and traffic. The prediction horizon ranges from 448 to 640, and the inference precision uses bfloat16. Regarding the hyperparameters of AAQM, we set the cutoff to 10 and the order to 5. During the evaluation phase, we separately counted the impact of AAQM and RD on Avg. Inference Time, Agg. Relative WQL, and Agg. Relative MASE, with other baseline methods using the same settings (Ansari et al. 2024).

As illustrated in Table 1, the accuracy enhancement provided by Apollo-Forecast becomes increasingly evident with longer prediction horizons (Tiny models are excluded due to negligible effects). In the best cases, it achieves a reduction in Agg. Relative WQL by up to 35.41% and in Agg. Relative MASE by up to 17.01%. In terms of inference speed, RD significantly improves performance by increasing the average inference time by as much as 2.7X. These improvements are particularly pronounced in scenarios involving long horizons and large models.

Public Dataset In this experiment, we selected seven types of records as input: *ETTh1*, *ETTh2*, *ETTm1*, *ETTm2*, *Weather*, *ECL*, and *TrafficL* (Zhou et al. 2021). The prediction horizon range is equal to the length of the test set, and the inference precision uses bfloat16. Regarding the hyperparameters of AAQM, we set the cutoff to 50 and the order to 5. During the evaluation phase, we measured the impact of AAQM and RD on inference time (minutes), WQL, and MASE with other baseline methods using the same settings.

The results are shown in Table 2 and 3. The accuracy improvement brought by Apollo-Forecast in zero-shot scenarios is very significant, with the highest reductions in WQL and MASE being 31.44% and 18.99%, respectively, compared to the baseline. In terms of inference speed, our approach can accelerate average inference time by up to 1.9X.

LBS Dataset In this experiment, we utilized real-time pedestrian flow data from a specific area in Shanghai, spanning from January 1, 2023, to July 1, 2024. The data, recorded at 5-minute intervals, includes timestamps and values. We fixed the prediction horizon at 100 and used bfloat16 for inference precision. Both Apollo-Forecast and Chronos models were set to the small size. For AAQM hyperparameters, we chose a cutoff of 30 and an order of 3. Since the RD mechanism was not employed in this experiment, we focused solely on the impact of AAQM on metrics, applying the same settings to other baseline methods.

As illustrated in Figure 3 and Table 4, Apollo-Forecast significantly enhances accuracy in zero-shot scenarios, reducing WQL and MASE by up to 31.53% and 14.29%, respectively, compared to Chronos. These results demonstrate the robustness of the proposed method in pedestrian flow prediction, achieving substantial accuracy improvements without a significant increase in computational costs.

Ablation Study In our ablation study, we aimed to verify the impact of different components within Apollo-Forecast. Using the UCR dataset and setting the horizon range from 512 to 640, we conducted experiments with evaluation methods consistent with previous sections. We tested two variant implementations: w/o-RD, where the RD mechanism in the Apollo-Forecast framework was disabled, and w/o-AAQM, where AAQM was removed from the framework, thus no longer attenuating high-frequency noise.

Table 5 shows the experimental results as follows:

1. **RD**: Disabling RD (w/o-RD) slightly improves WQL but decreases MASE marginally.
2. **AAQM**: Removing AAQM (w/o-AAQM) significantly worsens both WQL and MASE.

Methods	Apollo-S		Apollo-B		Apollo-L		Chronos-S		Chronos-B		Chronos-L	
Metric	WQL	MASE	WQL	MASE	WQL	MASE	WQL	MASE	WQL	MASE	WQL	MASE
ETTh1	2.135	4.120	**0.265**	3.869	0.267	4.049	2.309	4.421	0.307	**3.849**	0.339	4.276
ETTh2	2.124	4.107	**0.255**	**3.704**	0.286	4.100	2.321	4.414	0.287	3.883	0.332	4.263
ETTm1	2.126	4.120	0.270	**3.702**	**0.261**	4.149	2.327	4.418	0.292	3.926	0.336	4.327
ETTm2	2.135	4.102	**0.257**	**3.698**	0.291	4.081	2.323	4.446	0.287	4.073	0.334	4.138
Weather	2.135	4.129	**0.235**	**3.655**	0.292	3.697	2.298	4.390	0.343	3.907	0.350	3.967
ECL	2.163	4.105	**0.217**	**3.741**	0.255	3.785	2.306	4.442	0.292	4.163	0.284	4.672
TrafficL	2.092	4.094	**0.252**	4.022	0.308	4.279	2.286	4.411	0.292	**3.920**	0.335	4.248

Table 2: The performance of Apollo-Forecast and Chronos on public datasets.

Dataset	Apollo-S	Apollo-B	Apollo-L	Chronos-S	Chronos-B	Chronos-L
ETTh1	8.479 ± 0.122	9.866 ± 0.181	19.904 ± 0.243	10.758 ± 0.139	19.252 ± 0.193	36.488 ± 0.253
ETTh2	8.371 ± 0.117	9.853 ± 0.170	20.093 ± 0.237	10.569 ± 0.127	19.151 ± 0.185	36.429 ± 0.262
ETTm1	8.489 ± 0.123	9.810 ± 0.174	20.021 ± 0.234	10.820 ± 0.134	19.211 ± 0.195	36.477 ± 0.245
ETTm2	8.235 ± 0.115	9.734 ± 0.188	20.019 ± 0.224	10.819 ± 0.121	19.129 ± 0.184	36.488 ± 0.255
Weather	8.249 ± 0.131	9.866 ± 0.162	19.859 ± 0.250	10.501 ± 0.140	19.061 ± 0.171	36.362 ± 0.261
ECL	8.469 ± 0.105	9.965 ± 0.193	20.098 ± 0.245	10.801 ± 0.115	19.377 ± 0.185	36.621 ± 0.277
TrafficL	8.297 ± 0.136	9.996 ± 0.164	19.935 ± 0.223	10.702 ± 0.142	19.219 ± 0.170	36.495 ± 0.231

Table 3: The inference time (minutes) of Apollo-Forecast and Chronos on public datasets.

Methods	MASE	MAE	MSE	WQL
Apollo-S	**0.24**	**63.44**	**0.10**	**0.10**
Chronos-S	0.28	76.37	0.30	0.30
N-HiTS	0.37	99.48	0.72	0.60
N-BEATS	0.40	107.37	0.99	0.93
TFT	0.39	105.25	0.91	0.82
LSTM	0.42	113.24	1.00	1.00

Table 4: Performance of different algorithms on LBS. The values of MSE and WQL are scaled.

Horizon	512		576		640	
Metric	WQL	MASE	WQL	MASE	WQL	MASE
Apollo-M	5.98	17.15	4.41	17.11	6.23	18.07
w/o-RD	5.82	16.83	4.35	17.01	6.19	17.98
w/o-AAQM	9.31	18.53	6.05	20.50	9.40	21.61
Chronos-M	9.34	18.46	6.84	20.61	9.47	21.71

Table 5: Ablation analysis of Apollo-Forecast on the UCR dataset. WQL denotes aggregate relative WQL, and MASE is aggregate relative MASE.

In conclusion, both RD and AAQM are crucial for Apollo-Forecast's superior performance, with AAQM having a more significant impact on error reduction.

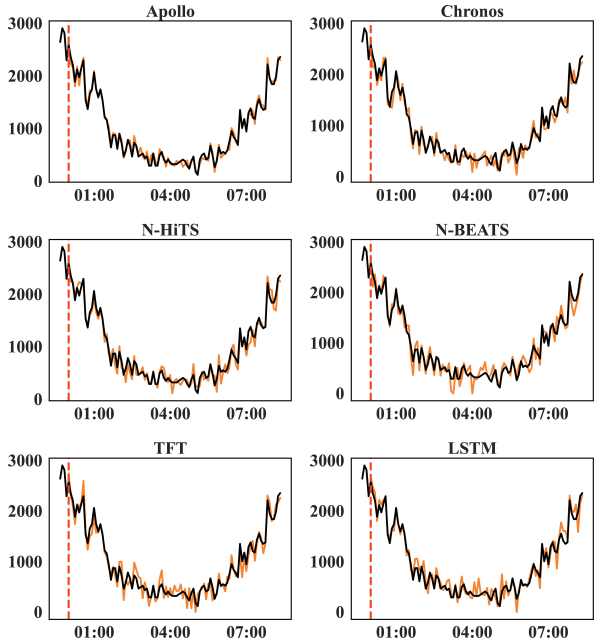

Figure 3: The results of our method and other benchmark models. The black line is the ground truth, the yellow line is the predicted value, and the red line is the start moment.

Conclusion

This study introduces the Apollo-Forecast approach, a novel framework designed to tackle aliasing distortion and slow inference in TSFM. Integrating the Anti-Aliasing Quantization Module effectively reduces distortion during tokenization. Additionally, Race Decoding boosts inference speed. Extensive experiments on diverse real-world datasets demonstrate its superior performance to SOTA methods.

Future work will focus on refining the framework further and exploring its applicability to other domains and tasks. We aim to enhance the generalization capability and efficiency of Apollo-Forecast in multi-frequency scenarios, making it a valuable tool for various applications, including finance, weather forecasting, and manufacturing.

Acknowledgments

This work was partly supported by the National Natural Science Foundation of China under grants 72374154 and 62273261.

References

Afrin, T.; and Yodo, N. 2022. A Long Short-Term Memory-based correlated traffic data prediction framework. *Knowledge-Based Systems*, 237: 107755.

Ansari, A. F.; Stella, L.; Turkmen, C.; Zhang, X.; Mercado, P.; Shen, H.; Shchur, O.; Rangapuram, S. S.; Arango, S. P.; Kapoor, S.; et al. 2024. Chronos: Learning the language of time series. arXiv:2403.07815.

Cai, W.; Liang, Y.; Liu, X.; Feng, J.; and Wu, Y. 2024. Msgnet: Learning multi-scale inter-series correlations for multivariate time series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 11141–11149. Washington, DC, USA.: AAAI Press.

Challu, C.; Olivares, K. G.; Oreshkin, B. N.; Ramirez, F. G.; Canseco, M. M.; and Dubrawski, A. 2023. Nhits: Neural hierarchical interpolation for time series forecasting. In *Proceedings of the AAAI conference on Artificial Intelligence*, 6989–6997. Washington DC, USA.: AAAI Press.

Chen, Z.; Ma, M.; Li, T.; Wang, H.; and Li, C. 2023. Long sequence time-series forecasting with deep learning: A survey. *Information Fusion*, 97: 101819.

Chowdhury, R. R.; Zhang, X.; Shang, J.; Gupta, R. K.; and Hong, D. 2022. Tarnet: Task-aware reconstruction for time-series transformer. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 212–220. Washington, DC, USA.: Association for Computing Machinery.

Cleaveland, M.; Lee, I.; Pappas, G. J.; and Lindemann, L. 2024. Conformal prediction regions for time series using linear complementarity programming. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 20984–20992. Washington, DC, USA.: AAAI Press.

Das, A.; Kong, W.; Sen, R.; and Zhou, Y. 2023. A decoder-only foundation model for time-series forecasting. arXiv:2310.10688.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305.

Deng, J.; Chen, X.; Jiang, R.; Song, X.; and Tsang, I. W. 2022. A multi-view multi-task learning framework for multi-variate time series forecasting. *IEEE Transactions on Knowledge and Data Engineering*, 35(8): 7665–7680.

Du, L.; Gao, R.; Suganthan, P. N.; and Wang, D. Z. 2022. Bayesian optimization based dynamic ensemble for time series forecasting. *Information Sciences*, 591: 155–175.

He, F.; Fu, T.-Y.; and Lee, W.-C. 2022. Rel-CNN: learning relationship features in time series for classification. *IEEE Transactions on Knowledge and Data Engineering*, 35(7): 7412–7426.

He, H.; Zhang, Q.; Wang, S.; Yi, K.; Niu, Z.; and Cao, L. 2023. Learning informative representation for fairness-aware multivariate time-series forecasting: A group-based perspective. *IEEE Transactions on Knowledge and Data Engineering*, 36(6): 2504 – 2516.

Jin, M.; Tang, H.; Zhang, C.; Yu, Q.; Liu, C.; Zhu, S.; Zhang, Y.; and Du, M. 2024a. Time series forecasting with llms: Understanding and enhancing model capabilities. arXiv:2402.10835.

Jin, M.; Wang, S.; Ma, L.; Chu, Z.; Zhang, J. Y.; Shi, X.; Chen, P.-Y.; Liang, Y.; Li, Y.-F.; Pan, S.; et al. 2023. Time-llm: Time series forecasting by reprogramming large language models. arXiv:2310.01728.

Jin, M.; Zhang, Y.; Chen, W.; Zhang, K.; Liang, Y.; Yang, B.; Wang, J.; Pan, S.; and Wen, Q. 2024b. Position paper: What can large language models tell us about time series analysis. arXiv:2402.02713.

Leviathan, Y.; Kalman, M.; and Matias, Y. 2023. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 19274–19286. PMLR.

Li, G.; and Jung, J. J. 2023. Deep learning for anomaly detection in multivariate time series: Approaches, applications, and challenges. *Information Fusion*, 91: 93–102.

Li, Y.; Wu, K.; and Liu, J. 2023. Self-paced ARIMA for robust time series prediction. *Knowledge-Based Systems*, 269: 110489.

Ma, Z.; Liu, P.; Choi, J.; and Sohn, H. 2024. High-sampled structural displacement estimation through the FIR filter-based two-stage fusion of high-sampled acceleration and temporally aliased low-sampled displacement measurements. *Mechanical Systems and Signal Processing*, 208: 111056.

Nate Gruver, S. Q., Marc Finzi; and Wilson, A. G. 2023. Large Language Models Are Zero Shot Time Series Forecasters. In *Advances in Neural Information Processing Systems*.

Rasul, K.; Ashok, A.; Williams, A. R.; Ghonia, H.; Bhagwatkar, R.; Khorasani, A.; Bayazi, M. J. D.; Adamopoulos, G.; Riachi, R.; Hassen, N.; Biloš, M.; Garg, S.; Schneider, A.; Chapados, N.; Drouin, A.; Zantedeschi, V.; Nevmyvaka, Y.; and Rish, I. 2024. Lag-Llama: Towards Foundation Models for Probabilistic Time Series Forecasting. arXiv:2310.08278.

Tang, Y.; Song, Z.; Zhu, Y.; Yuan, H.; Hou, M.; Ji, J.; Tang, C.; and Li, J. 2022. A survey on machine learning models for financial time series forecasting. *Neurocomputing*, 512: 363–380.

Vukovic, D. B.; Romanyuk, K.; Ivashchenko, S.; and Grigorieva, E. M. 2022. Are CDS spreads predictable during the Covid-19 pandemic? Forecasting based on SVM, GMDH, LSTM and Markov switching autoregression. *Expert systems with applications*, 194: 116553.

Wang, H.; Wang, J.; Zhao, Y.; Liu, Q.; Liu, M.; and Shen, W. 2022. Few-shot learning for fault diagnosis with a dual graph neural network. *IEEE Transactions on Industrial Informatics*, 19(2): 1559–1568.

Zeng, A.; Chen, M.; Zhang, L.; and Xu, Q. 2023. Are Transformers Effective for Time Series Forecasting? In *Proceedings of the AAAI conference on Artificial Intelligence*, 11121–11128. Washington DC, USA.: AAAI Press.

Zhang, X.; Chowdhury, R. R.; Gupta, R. K.; and Shang, J. 2024. Large language models for time series: A survey. arXiv:2402.01801.

Zhang, Z.; Chen, Y.; Zhang, D.; Qian, Y.; and Wang, H. 2023. Ctfnet: Long-sequence time-series forecasting based on convolution and time–frequency analysis. *IEEE Transactions on Neural Networks and Learning Systems*, 1–15.

Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI conference on artificial intelligence*, 11106–11115. Palo Alto, California USA.: AAAI Press.