TimeTok: Granularity-Controllable Time-Series Generation via Hierarchical Tokenization

Seokhyun Lee[1]* **Jaeho Kim**[1]* **Changjun Oh**[1]
Mihaela van der Schaar[2] **Changhee Lee**[1]†
[1]Department of Artificial Intelligence, Korea University, Seoul, South Korea
[2]Department of Applied Mathematics and Theoretical Physics,
University of Cambridge, Cambridge, United Kingdom

Abstract

Time-series data are inherently multiscale, spanning diverse temporal granularities from coarse trends to fine-scale dynamics. However, existing time-series generative models provide limited control over the temporal granularity of both inputs and outputs, restricting their ability to condition on user-provided coarse sketches and generate samples at a desired target granularity. To address this, we introduce TimeTok, a unified framework for *Granularity-Controllable Time-Series Generation* (GC-TSG), which generates time series at *any* target granularity from *any* coarser input (*e.g.,* rough sketches) or without conditioning. At the core of TimeTok is a hierarchical tokenization strategy that maps time series into an ordered sequence of tokens, from coarse to fine temporal granularity. Our autoregressive generation process operates across these granularity levels, producing token blocks that are decoded back into continuous time series. This design naturally enables GC-TSG within a single framework, where controlling the number of token blocks provides explicit control over output detail. Experiments show that TimeTok excels at GC-TSG tasks while achieving state-of-the-art performance in standard generation. Furthermore, we showcase TimeTok's potential as a foundational tokenizer by training on multiple datasets with heterogeneous temporal granularities, verifying strong transferability that consistently outperforms models trained on individual datasets. To our knowledge, this is the first unified framework that covers the full generative spectrum for time series, offering a valuable foundation for models that benefit from diverse temporal granularities. Codes are available here.

1 Introduction

Time-series data are inherently multiscale: they can be observed, represented, and analyzed at different levels of temporal granularities. Coarse-grained representations capture slowly varying trends and long-range dependencies, whereas fine-grained representations preserve rapid fluctuations and detailed dynamics. In practice, however, acquiring fine-grained time-series data is often costly or impractical due to hardware costs, storage limitations, and privacy concerns. As a result, there is a persistent granularity mismatch between the coarse-grained data typically available and the fine-grained information required for reliable downstream modeling and decision-making.

This mismatch exposes a limitation of the standard formulation of time-series generation. Existing models typically learn to sample time series at the single temporal granularity present in the training data, but they do not provide a native interface for asking: given this coarse observation, what fine-grained dynamics are plausible? Or, given no observation, can we generate directly at a user-specified level of temporal detail? Thus, the missing capability is not only better sample quality, but also *controllability over the temporal granularity of both the conditioning input and the generated output.*

*Equal contribution.
†Corresponding author: changheelee@korea.ac.kr

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

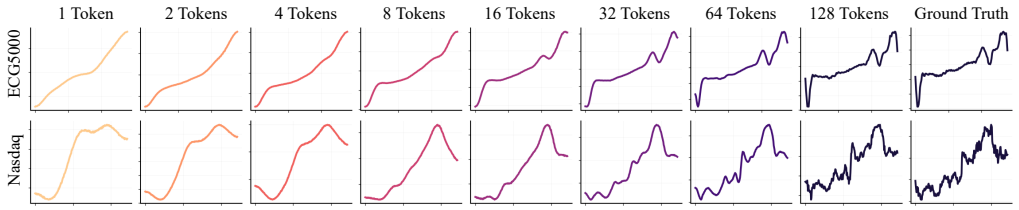


Figure 1: **Visualization at Different Granularity.** `TimeTok` encodes time series into a hierarchically ordered sequence of discrete tokens that progressively capture coarse-to-fine temporal granularities. We visualize samples from the medical (*i.e.,* ECG) and finance domain (*i.e.,* Nasdaq stock data); additional results in Appendix E.

This challenge is particularly consequential in high-stakes domains such as healthcare and finance, where decision-making and data analysis routinely span multiple temporal granularities. In healthcare, hospital-grade electrocardiogram (ECG) signals are recorded at high sampling rates (up to 1000 Hz) but consumer-grade smartwatches collect ECG data at much lower sampling rates (often below 60 Hz) [1]. In finance, market data is recorded at a millisecond level but analyzed at task-dependent granularities, ranging from high-frequency trading [2] to monthly portfolio rebalancing [3], yet existing time-series generation methods provide limited support for bridging across these scales.

We therefore study granularity-controllable time-series generation (GC-TSG): the problem of learning *one generative model that can synthesize time series at any target granularity, either unconditionally or conditioned on an input observed at any coarser granularity.* Standard time-series generation is then only one special case, corresponding to generation from scratch at the finest granularity. Conditional refinement, such as transforming a low-fidelity ECG or a user-provided stock-market sketch into a plausible fine-grained trajectory, is another case of the same problem. This framing turns temporal granularity from a nuisance of preprocessing into an explicit generative control variable. The central technical question is therefore *how can we build a time-series representation in which temporal granularity is ordered, controllable, and generative.*

To answer this, we propose `TimeTok`, a unified framework for granularity-controllable time-series generation. At the core of `TimeTok` is a hierarchical tokenizer that represents a time series as an ordered sequence of discrete tokens, where early tokens encode coarse temporal structure and later tokens add progressively finer dynamics. This gives token length a semantic meaning: choosing how many tokens to decode controls the temporal granularity of the output. Building on this representation, `TimeTok` uses block-wise autoregressive generation across granularity levels, generating the next block of finer-grained tokens conditioned on the coarser prefix. The resulting model can generate from scratch at any granularity, refine coarse observations into finer trajectories, and use user-provided sketches as conditioning inputs, all within the same architecture.

Contribution. (i) We formulate GC-TSG as a unified generative problem over pairs of temporal granularities, making standard generation and conditional temporal refinement instances of the same task. (ii) We introduce `TimeTok`, a hierarchical tokenization and autoregressive generation framework in which temporal granularity is controlled directly through the number of generated token blocks. (iii) We show empirically that this representation is not only controllable but useful: `TimeTok` outperforms strong time-series generation baselines on GC-TSG, achieves state-of-the-art or competitive standard generation performance, improves downstream prediction under granularity mismatch, and scales to a foundation-tokenizer trained on heterogeneous multi-domain time-series data.

2 Related Works

`TimeTok` connects three lines of work that have been mostly discussed separately: time-series generation, time-series tokenization, and coarse-to-fine token generation. Its novelty is not any one component in isolation, but the alignment of these three components into a single, unified generative interface for GC-TSG. The hierarchical tokenizer gives token prefixes a temporal-granularity semantics; the block-wise autoregressive model learns a refinement trajectory across those prefixes; and the inference procedure allows arbitrary coarse inputs, including low-fidelity observations and user sketches, to be refined at a user-defined granularity. This enables `TimeTok` to support standard generation, unconditional generation towards any intermediate granularities, and conditional coarse-to-fine

Table 1: Contributions of TimeTok relative to prior works. More detailed comparisons are provided in Table 9.

Line of work	Methods	Main capability	Limitation in GC-TSG/ Relation to TimeTok
Standard time-series generation	TimeGAN [4], TSGDiff [5], TimeVQVAE [6], Diffusion-TS [7], ImagenTime [8]	Generates quality samples at the *trained* granularity.	These models do not natively expose temporal granularity as a controllable dimension or model transitions between granularities. TimeTok includes Standard time-series generation as a special case, but also supports conditional refinement and generation at any target granularity
Time-series tokenization	TOTEM [9]	Learns general-purpose token representations for time-series tasks.	Token representations from these models are unordered. TimeTok imposes an ordered temporal hierarchy over tokens, enabling controllable granularity.
Multi-granular tokens	WaveToken [10], TransPL [11]	Captures frequency-aware and trend–residual temporal structure.	Granularity is usually fixed by the architecture rather than specified by the user at inference time. TimeTok turns granularity into a generative control variable.
Hierarchical image generation	VAR [12], FlowAR [13], FlexTok [14]	Uses ordered or variable-length token generation to improve image synthesis quality and efficiency.	Intermediate scales are mainly latent scaffolds for the final-resolution image, and variable prefixes control the complexity of generated image rather than modeling refinement dynamics or conditional generation across granularity pairs. TimeTok instead treats granularity as user-facing input-output control for conditional refinement.
TimeTok	**This work**	**Hierarchical temporal tokenization with block-wise autoregressive coarse-to-fine generation.**	**TimeTok is the first unified framework that enables: (i) time series generation from scratch, (ii) generation at intermediate granularities, and (iii) conditional refinement from coarse inputs.**

refinement, all within a single model. We summarize the contributions of TimeTok relative to prior works in Table 1 and provide a more detailed discussion of the related works in Appendix C.

3 Modeling Time-Series Generation Across Temporal Granularities

In this work, we posit that a time-series generative model should treat *granularity* as a controllable dimension, enabling traversal across granularity levels during generation.

3.1 Generation with Multiple Temporal Granularity

Let $\mathbf{x} = [x_1, \ldots, x_T] \in \mathbb{R}^T$ be a univariate time series of length T, drawn from an unknown data distribution p_{data}, *i.e.,* $\mathbf{x} \sim p_{\text{data}}(\mathbf{x})$. We formally define granularity level $\ell \in \{1, \ldots, L\}$ as a specification of how much temporal detail is preserved in a given time series. For each time series $\mathbf{x}$, we denote $\mathbf{x}^{(\ell)}$ as a time series with temporal granularity level ℓ where each $\mathbf{x}^{(\ell)}$ shares the same high-level semantics as $\mathbf{x}$ but differs in its temporal details; a smaller value corresponds to coarser time series, while a larger value preserves more fine-grained temporal information. By convention, we set $\mathbf{x}^{(L)} \stackrel{\text{def}}{=} \mathbf{x}$, corresponding to the finest granularity and indicating the original time series.

Standard Time-series Generation. The *Standard time-series generation* focuses on learning a density p_θ whose samples are indistinguishable from those drawn from the marginal data distribution p_{data}, achieving $p_\theta(\mathbf{x}) \stackrel{\text{dist.}}{=} p_{\text{data}}(\mathbf{x})$, where $\stackrel{\text{dist.}}{=}$ denotes distributional equality. While fundamental, it offers no control over the temporal detail of outputs or conditioning on high-level temporal structures.

Granularity-Controllable Time-series Generation. We define GC-TSG as a generalized time-series generation task with explicit control over temporal granularity for both conditioning and synthesis as follows:

> **Definition (GC-TSG).** Let $\mathbf{x}^{(i)}$ and $\mathbf{x}^{(j)}$ denote time-series representations at temporal granularity levels $i, j \in \{0, 1, \ldots, L\}$ with $j > i$. We define *granularity-controllable time-series generation (GC-TSG)* as the task of learning a generative model p_θ whose samples are indistinguishable from those drawn from the conditional data distribution across all valid pairs of temporal granularities, achieving $p_\theta(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)}) \stackrel{\text{dist.}}{=} p_{\text{data}}(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)})$. Here, we slightly abuse notation by defining $\mathbf{x}^{(0)} = \varnothing$, which corresponds to the unconditioned setting.

This formulation unifies unconditional generation (*i.e.,* $i = 0$) and conditional refinement (*i.e.,* $i \geq 1$) under a single framework. Standard generation corresponds to the special case $i = 0$ and $j = L$.

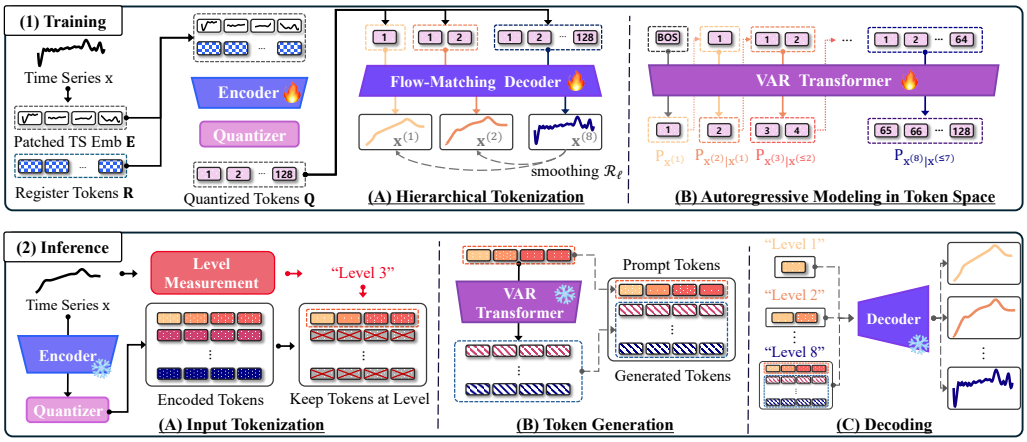


Figure 2: **An Overview of `TimeTok`. (1) Training:** The hierarchical tokenizer is trained to encode input time series into a sequence of discrete tokens structured in a coarse-to-fine temporal hierarchy. These tokens are then used to train the VAR Transformer that autoregressively predicts tokens at the next granularity level. **(2) Inference:** A conditioning time series of unknown granularity is encoded into a full token sequence, from which tokens at the inferred granularity are selected as a prefix for the VAR Transformer to generate the remaining tokens. The completed token sequence is then decoded into a time series at the target granularity.

3.2 Autoregressive Formulation of GC-TSG

In practice, time-series datasets typically provide only the finest-grained observations $\mathbf{x}^{(L)}$. To construct the multi-granular hierarchy required for granularity-controllable time-series generation in a principled manner, we introduce a family of *information-reducing operators* $\{\mathcal{R}_\ell\}_{\ell\in[L]}$. Each operator $\mathcal{R}_\ell$ maps the raw time series to its representation at temporal granularity level ℓ, *i.e.,* $\mathbf{x}^{(\ell)} = \mathcal{R}_\ell(\mathbf{x})$. Here, smaller values of ℓ induce stronger information reduction, resulting in coarser abstractions of the time series. For instance, $\mathcal{R}_\ell$ can be implemented as a Gaussian smoothing kernel [15] or a moving-average operator, where ℓ controls the smoothing variance or the window size, respectively, to determine the temporal granularity[3].

A core objective of GC-TSG is to model conditional distributions over time-series representations for any valid pair of temporal granularities, using only the finest-grained time series (*i.e.,* observations). To achieve this, we decompose the marginal log-likelihood of the observed time series by applying the chain rule along the granularity axis:

$$\log p_\theta(\mathbf{x}) = \log p_\theta\left(\mathbf{x}^{(\le L)}\right) = \sum_{\ell=1}^{L} \log p_\theta\left(\mathbf{x}^{(\ell)} | \mathbf{x}^{(<\ell)}\right) \tag{1}$$

where $\mathbf{x}^{(<\ell)} = \left(\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\ell-1)}\right)$. The first equality holds because each coarse-granular representation $\mathbf{x}^{(\ell)}$ is deterministically obtained from $\mathbf{x}$ via $\mathcal{R}_\ell$. This hierarchical autoregressive decomposition induces a joint distribution over representations at all temporal granularities and, in doing so, defines a valid *refinement process* that incrementally introduces finer temporal details.

Under this formulation, generating $\mathbf{x}^{(j)}$ conditioned on $\mathbf{x}^{(i)}$ is realized by sampling a coarse-to-fine sub-trajectory. Conditioning on a prefix consistent with $\mathbf{x}^{(i)}$, ancestral sampling proceeds through successive conditional distributions $p_\theta\left(\mathbf{x}^{(\ell)} \mid \mathbf{x}^{(<\ell)}\right)$ for $\ell = i+1, \ldots, j$, progressively refining the time-series representations until reaching granularity level j. This procedure ensures that the generated $\mathbf{x}^{(j)}$ remains both statistically consistent with the conditioning input $\mathbf{x}^{(i)}$ and coherent with the hierarchical structure.

Unfortunately, learning and sampling from this hierarchical formulation in Eq. (1) present two key challenges: (i) Implementing each refinement step – *i.e.,* $p_\theta(\mathbf{x}^{(\ell)}|\mathbf{x}^{(<\ell)})$ – directly in the raw time-series space is inherently difficult, as it requires conditioning on the entire trajectory of preceding coarser time series, which makes training computationally expensive and often unstable, especially for long time series. (ii) Conditional generation given $\mathbf{x}^{(i)}$ implicitly assumes access to the full coarse-level prefix $\mathbf{x}^{(\le i)}$, which is typically unavailable at inference time and otherwise needs to be reconstructed. These challenges motivate hierarchical modeling in a structured discrete token space.

[3] In practice, the degree of temporal granularity can be quantified using established measures in time series, such as the detrended fluctuation analysis (DFA) exponent [16].

4 Method: TimeTok

To address the challenge of modeling and generating time series with hierarchical dependencies across temporal granularities, we introduce TimeTok, a novel framework for GC-TSG comprising three key components; an overview of TimeTok is illustrated in Figure 2:

- **Hierarchical Tokenization.** Continuous time series are mapped into a sequence of discrete tokens that encode temporal structure in a coarse-to-fine hierarchy.
- **Autoregressive Modeling in Token Space.** Generative modeling is then performed in the learned discrete token space using a hierarchical autoregressive model.
- **Granularity-Controllable Inference.** At inference, coarse time-series structure is encoded into a token prefix, and additional tokens are autoregressively generated up to a target granularity. The resulting token sequence is decoded to produce a time-series at the desired temporal granularity.

4.1 Hierarchical Tokenization for Granularity Control

To encode a coarse-to-fine temporal hierarchy directly in the discrete latent space, we introduce a structured tokenization in which the token ordering is semantically aligned with temporal granularity. Specifically, the leading tokens are constrained to represent high-level information of the time series, while successive tokens incrementally encode fine-scale temporal dynamics. In contrast to conventional tokenization schemes that require generating a fixed-length token sequence, our approach allows variable-length token sequences depending on the temporal granularity.

Discrete Representation via Register Tokens. Following the discrete representation learning paradigm [17, 18], we assume an encoder–decoder architecture $(\texttt{Enc}, \texttt{Dec})$ and a quantizer $\mathcal{Q}$. Given an input time series $\mathbf{x}$, we first partition it into W non-overlapping patches of equal length. Each patch is linearly projected into a latent space, producing a sequence of patch embeddings $\mathbf{E} = [\mathbf{e}_1, \ldots, \mathbf{e}_W] \in \mathbb{R}^{W \times h}$, where h denotes the hidden dimension. To obtain a compact and structured representation, we append M *learnable* register tokens $\mathbf{R} = [\mathbf{r}_1, \ldots, \mathbf{r}_M] \in \mathbb{R}^{M \times h}$ [19] to the patch embeddings, forming the augmented sequence $\mathbf{E} \oplus \mathbf{R} \in \mathbb{R}^{(W+M) \times h}$, where $\oplus$ denotes concatenation. This sequence is then processed by a Transformer-based encoder (Enc). We denote the encoder outputs corresponding to the register tokens as $\widetilde{\mathbf{R}} = [\tilde{\mathbf{r}}_1, \ldots, \tilde{\mathbf{r}}_M]$. Each encoded register token is subsequently discretized by finite scalar quantization (FSQ) [18], resulting in a sequence of discrete latent codes. In our implementation, we configure the FSQ codebook with levels `[4,4,4,4,4,4]`, which maps each register token to a 6-dimensional discrete code, inducing an effective vocabulary size of 4^6. The resulting quantized register tokens $\mathbf{Q} = [\mathbf{q}_1, \ldots, \mathbf{q}_M] = [\mathcal{Q}(\tilde{\mathbf{r}}_1), \ldots, \mathcal{Q}(\tilde{\mathbf{r}}_M)] \in \mathbb{R}^{M \times 6}$ are used as the conditioning input to the decoder.

Token Conditioned Flow Matching Decoder. To enforce the desired hierarchical structure, we design the decoder (Dec) using a *Conditional Flow Matching* framework [20]. Flow Matching generates samples by continuously transforming a source distribution (*e.g.,* Gaussian noise) into the data distribution via a time-dependent velocity field. Our key idea is to endow the quantized register tokens with explicit hierarchical semantics by training the decoder to reconstruct time-series representations at different temporal granularities, depending on how many tokens are provided.

For each target granularity level ℓ, we define n_ℓ as the number of tokens required to represent the time series at that level, with $n_L = M$. In practice, we set $n_\ell = 2^{\ell-1}$ and fix the maximum granularity level to $L = 8$. Empirically, we observe that uniform and exponential token allocations produce nearly identical losses across levels, indicating that fewer tokens suffice for coarser levels (Appendix D.3). The decoder is conditioned only on the n_ℓ leading tokens $\mathbf{Q}_{1:n_\ell} = [\mathbf{q}_1, \ldots, \mathbf{q}_{n_\ell}]$ and tasked with generating the corresponding time-series representation $\mathbf{x}^{(\ell)}$. Here, the target $\mathbf{x}^{(\ell)}$ is obtained from the original time-series $\mathbf{x}$ via the information-reducing operator $\mathcal{R}_\ell$, *i.e.,* $\mathbf{x}^{(\ell)} = \mathcal{R}_\ell(\mathbf{x})$.[4] In contrast to conventional decoders, which condition the velocity field on the full token sequence $\mathbf{Q}$, we train the velocity predictor $\mathbf{v}_\theta$ (parameterized by a neural network) under varying conditioning budgets, corresponding to different numbers of available tokens. Each conditioning budget induces a distinct generative trajectory: conditioning on fewer tokens encourages the model to capture coarse, high-level temporal structure, while additional tokens progressively refine generation with finer details.

[4] In our implementation, we instantiate $\mathcal{R}_\ell$ as a Gaussian smoothing operator that progressively removes fine-grained temporal details. See more details provided in Appendix A.1.

Formally, for each granularity level ℓ, we define the linear interpolation between a clean sample $\mathbf{x}^{(\ell)}$ and noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, I)$ at time $\tau \in [0, 1]$ as $\mathbf{x}_\tau^{(\ell)} = (1-\tau)\,\boldsymbol{\epsilon} + \tau\,\mathbf{x}^{(\ell)}$, with the corresponding target velocity $\mathbf{v}_\tau^{(\ell)} = \mathbf{x}^{(\ell)} - \boldsymbol{\epsilon}$. This velocity specifies the instantaneous direction that transports noise toward $\mathbf{x}^{(\ell)}$, and is approximated by the parameterized velocity $\mathbf{v}_\theta$ conditioned on $\mathbf{Q}_{1:n_\ell}$. We train $\mathbf{v}_\theta$ using the following coarse-to-fine flow-matching objective:

$$\mathcal{L}_{\mathrm{CTF}}(\theta) = \mathbb{E}_{\mathbf{x},\,\boldsymbol{\epsilon},\,\tau,\,\ell}\Big[\,\big\|\mathbf{v}_\theta(\mathbf{x}_\tau^{(\ell)} \,|\, \mathbf{Q}_{1:n_\ell}, \tau) - \mathbf{v}_\tau^{(\ell)}\big\|_2^2\,\Big], \tag{2}$$

where the granularity level ℓ is sampled uniformly during training. This objective explicitly enforces that conditioning on only the first n_ℓ tokens suffices to reconstruct the time-series representation with granularity level ℓ, thus inducing a coarse-to-fine semantic ordering over the token sequence.

4.2 GC-TSG through Autoregressive Transformer

Once the encoder and decoder $(\texttt{Enc}, \texttt{Dec})$ are trained under the objective in Eq. (2), we freeze these components and shift the generative task to the learned hierarchical token space. This allows us to reformulate the data log-likelihood, $\log p_\theta(\mathbf{x})$, in terms of the corresponding tokens using a block-wise autoregressive factorization: $\log p_\phi\big(\mathbf{Q}_{1:n_L}\big) = \sum_{\ell=1}^{L} \log p_\phi\big(\mathbf{Q}_{n_{\ell-1}+1:n_\ell} \,\big|\, \mathbf{Q}_{1:n_{\ell-1}}\big)$ where the conditional distributions are parameterized by ϕ and implemented using a VAR Transformer [12]. Unlike conventional autoregressive models that generate sequences via token-by-token, next-token prediction, our VAR Transformer operates at the level of granularity blocks. It predicts a group of tokens corresponding to the next temporal granularity level, conditioned on all preceding coarser-level tokens as a prefix. We train p_ϕ using the following block-wise autoregressive objective:

$$\mathcal{L}(\phi) = -\mathbb{E}_{\mathbf{Q}}\left[\sum_{\ell=1}^{L} \log p_\phi\big(\mathbf{Q}_{n_{\ell-1}+1:n_\ell} \,\big|\, \mathbf{Q}_{1:n_{\ell-1}}\big)\right], \tag{3}$$

where $\mathbf{Q}$ is sampled from the tokenized time series in the training set. By sequentially generating token blocks across granularity levels, the model learns to refine coarse time-series representations with progressively finer details. This sequential block generation realizes Eq. (1) in the token space, capturing the full refinement trajectory with only M tokens – *i.e.,* the number of tokens required to represent a finest-grained time series – that enable efficient GC-TSG.

4.3 Granularity-Controllable Generation (Inference)

The key capability of `TimeTok` is its ability to generate fine-grained time series from coarse inputs, such as low-fidelity series or user-provided sketches. Given a source granularity level i and a target level j, `TimeTok` performs GC-TSG through the following three steps:

① **Input Tokenization.** Since the input granularity is not known a priori, we first encode the input time series into a sequence of tokens $\mathbf{Q}$. We then determine its granularity level i by evaluating the reconstruction error at each level (detailed in Appendix A.3). The prefix $\mathbf{Q}_{1:n_i}$, corresponding to the inferred level i, is then extracted to serve as the initial conditioning for the VAR Transformer.

② **Token Generation.** Using $\mathbf{Q}_{1:n_i}$ as the initial context, we generate tokens in a coarse-to-fine manner. For each successive level $\ell = i{+}1, \ldots, j$, we sample the corresponding token block: $\hat{\mathbf{Q}}_{n_{\ell-1}+1:n_\ell} \sim p_\phi\big(\cdot \,\big|\, \mathbf{Q}_{1:n_i}, \hat{\mathbf{Q}}_{n_i+1:n_{\ell-1}}\big)$ conditioning on both the original prefix and all previously generated tokens (with $\hat{\mathbf{Q}}_{n_i+1:n_{\ell-1}} = \varnothing$ for $\ell = i{+}1$). This iterative process continues until we obtain the complete token sequence with granularity level j.

③ **Decoding.** The resulting token sequence is decoded back to the time-series representation at granularity level j, *i.e.,* $\hat{\mathbf{x}}^{(j)} = \texttt{DEC}\big(\mathbf{Q}_{1:n_i} \oplus \hat{\mathbf{Q}}_{n_i+1:n_j}\big)$.

Generation from scratch is a special case in which no coarse input is provided. In this setting, the conditioning context reduces to the `[BOS]` token, from which we autoregressively generate n_j tokens. Decoding the resulting tokens gives a time series at granularity level j. Similarly, generating all n_L tokens becomes standard time-series generation. In both conditional and unconditional settings, class conditioning can be incorporated via AdaLN [21].

5 Experiments

We provide the details of the datasets and experiments. Full implementation details, evaluation metrics, and ablation study can be found in Appendix A and D.3, respectively.

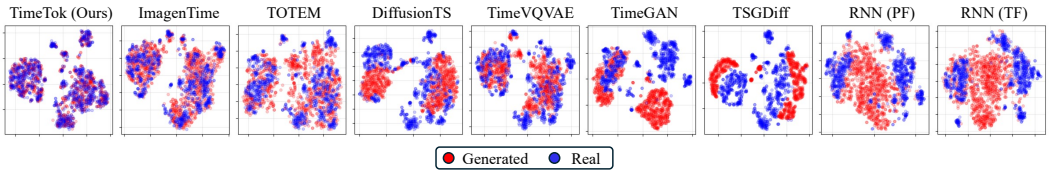


Figure 3: **t-SNE visualization of ECG5000**. We observe a higher overlap between the synthetic and real data distribution for `TimeTok`, while other methods show signs of mode collapse.

Datasets & Generation Tasks. We evaluate `TimeTok` on two classification datasets (ECG5000 and ItalyPowerDemand) and two forecasting datasets (Nasdaq and ETTh1). For standard generation, where time series are generated from scratch (`[BOS]` token), we measure the quality and distributional similarity between the synthetic and real data. In GC-TSG, we generate $\hat{\mathbf{x}}^{(j)}$ from $\mathbf{x}^{(i)}$ ($j > i > 0$), and evaluate (i) the diversity of the generated sequences $\hat{\mathbf{x}}^{(j)}$ and its alignment with $\mathbf{x}^{(j)}$, and (ii) structural consistency between $\mathbf{x}^{(i)}$ and $\hat{\mathbf{x}}^{(j)}$.

Evaluation Metrics. In standard generation, we evaluate whether the generated time series aligns with real data. To assess this alignment, we measure (i) predictive score [4] – *i.e.,* the downstream classification or forecasting performance in a Train-Synthetic-Test-Real (TSTR) setup – and (ii) FID score – *i.e.,* distributional similarity between synthetic and real data, for which we adopt the FID evaluation module from DiffusionTS [7]. For GC-TSG, we evaluate the ability to generate diverse fine-grained time-series $\hat{\mathbf{x}}^{(j)}$ from coarse inputs $\mathbf{x}^{(j)}$, while preserving the structure of the coarse series. As such, we use CRPS_{sum} (Continuous Ranked Probability Score) [22], a commonly used metric for probabilistic forecasting, to measure diversity and consistency. We also measure the C-Cons (coarse consistency) between $\mathbf{x}^{(i)}$ and re-coarsened $\hat{\mathbf{x}}^{(j)}$ using RMSE.

Baseline Methods. We compare `TimeTok` against state-of-the-art methods including RNNs trained with teacher forcing (TF) [23], professor forcing (PF) [24], TSGDiff [5], TimeGAN [4], TimeVQ-VAE [6], DiffusionTS [7], TOTEM [9], and ImagenTime [8]. TSGDiff and ImagenTime are evaluated only for standard generation, as extending them to GC-TSG requires substantial architectural modifications; details on baseline selection are provided in Appendix A.6.

Extension to Foundational Tokenizer. So far, we have assumed that the granularity level of a given time-series dataset corresponds to the finest granularity level (*i.e.,* $\mathbf{x}^{(L)} \overset{\text{def}}{=} \mathbf{x}$). This assumption, however, may not hold in heterogeneous settings, such as foundation model training, where data naturally spans multiple temporal granularities, and uniformly treating all time series as finest-granular may be inappropriate. To address this gap, we extend `TimeTok` by determining the granularity level using DFA, rather than uniformly treating all samples as level L. We then train the hierarchical tokenizer on UTSD [25], a multi-domain time-series collection designed for foundation models. We evaluate the transferability of `TimeTok` under this setup; more details are provided in Section 5.3.

5.1 Standard Time-Series Generation

Quantitative Performance. For the TSTR setup, we train a backbone model (*i.e.,* InceptionTime [26] for classification, DLinear [27] for forecasting) and report its predictive performance. We provide class conditioning to all models used in classification. As shown in Table 2, `TimeTok` achieves the best performance on ECG5000, ItalyPowerDemand, and Nasdaq, and competitive performance on ETTh1, in both predictive and FID metrics. Here, Oracle indicates the predictive performance when the prediction models are trained on the real data.

Visualizations with t-SNE. Figure 3 shows t-SNE visualizations of the ECG5000 dataset. Synthetic data generated by `TimeTok` (red) strongly overlaps with the real data distribution (blue), indicating high diversity and fidelity. In contrast, baseline methods show evidence of mode collapse or incomplete coverage of the true distribution.

5.2 Granularity-Controllable Time-Series Generation

We evaluate `TimeTok` on GC-TSG with two related tasks: (i) *conditional* generation, where given a coarse input, each model generates diverse and finer-grained time series at a specified target granularity while preserving the coarse structure, and (ii) *unconditional* generation, where each model generates time series at a specified target granularity level from scratch. We provide the results in Table 3 and Table 4, respectively. For fair comparison, we adapt the baselines to support granularity

Table 2: **Standard Generation.** Generation performance on classification (ECG5000, ItalyPowerDemand; ACC/F1) and forecasting (Nasdaq, ETTh1; MSE) tasks, with FID reported across all datasets.

Method	ECG5000			ItalyPowerDemand			Nasdaq		ETTh1	
	FID ↓	ACC ↑	F1 ↑	FID ↓	ACC ↑	F1 ↑	FID ↓	MSE ↓	FID ↓	MSE ↓
Oracle	–	0.959±0.001	0.620±0.010	–	0.982±0.008	0.982±0.008	–	0.033±0.000	–	0.098±0.000
RNN(TF)	0.220±0.047	0.895±0.014	0.383±0.024	0.125±0.034	0.903±0.027	0.903±0.027	0.462±0.083	0.046±0.000	1.195±0.040	0.104±0.000
RNN(PF)	0.076±0.028	0.884±0.013	0.398±0.035	0.101±0.019	0.952±0.020	0.952±0.021	1.070±0.119	0.054±0.001	0.730±0.015	0.112±0.001
TSGDiff	0.102±0.033	0.854±0.006	0.467±0.047	0.088±0.011	0.931±0.027	0.931±0.028	0.786±0.085	0.070±0.000	1.185±0.150	0.111±0.000
TimeGAN	0.034±0.002	0.815±0.042	0.348±0.018	0.026±0.008	0.958±0.003	0.958±0.003	0.316±0.050	0.061±0.001	1.068±0.243	0.104±0.001
TimeVQVAE	0.033±0.011	0.925±0.004	0.380±0.001	0.060±0.011	0.979±0.003	0.979±0.003	0.136±0.015	0.048±0.001	0.107±0.009	0.102±0.001
DiffusionTS	0.041±0.009	0.896±0.015	0.552±0.028	0.023±0.005	0.979±0.003	0.979±0.003	0.208±0.015	0.056±0.001	0.083±0.011	0.101±0.001
TOTEM	0.022±0.006	0.923±0.004	0.379±0.001	0.016±0.004	0.979±0.007	0.979±0.007	0.164±0.008	0.046±0.001	0.102±0.015	0.102±0.001
ImagenTime	0.020±0.010	0.940±0.007	0.477±0.104	0.043±0.006	0.978±0.002	0.978±0.002	0.142±0.032	0.046±0.000	0.074±0.007	**0.099**±0.000
TimeTok	**0.004**±0.001	**0.948**±0.006	**0.564**±0.047	**0.007**±0.001	**0.982**±0.000	**0.982**±0.000	**0.091**±0.017	**0.045**±0.001	**0.070**±0.006	0.103±0.000

Table 3: **Conditional Generation.** Conditional GC-TSG performance averaged over all granularity pairs (Avg, $i \to j$, $0 < i < j$). C-Cons denotes coarse-level consistency. Full results of TimeTok are in Appendix D.

	Classification (ECG5000)		Forecasting (Nasdaq)	
Method	**CRPS_{sum} ↓**	**C-Cons ↓**	**CRPS_{sum} ↓**	**C-Cons ↓**
RNN(TF)	5.602±1.869	0.059±0.020	86.72±52.09	0.483±0.295
RNN(PF)	3.797±1.067	0.034±0.011	48.20±22.14	0.370±0.215
TimeGAN	1.737±0.737	0.011±0.005	39.11±22.14	0.196±0.105
TimeVQVAE	3.693±0.858	0.044±0.013	35.32±34.30	0.282±0.258
DiffusionTS	2.672±1.551	0.031±0.019	101.5±63.26	0.771±0.538
TOTEM	2.639±1.286	0.035±0.019	35.97±22.54	0.294±0.247
TimeTok	**1.587**±0.867	**0.010**±0.006	**15.87**±12.48	**0.096**±0.090

Table 4: **Unconditional Generation.** Unconditional generation to each granularity level, averaged over all levels. Full results are provided in Appendix D.

	ECG5000	Nasdaq
Method	**FID ↓**	**FID ↓**
RNN(TF)	0.884±0.316	8.339±0.856
RNN(PF)	0.251±0.047	4.199±0.351
TimeGAN	0.788±0.194	9.568±0.658
TimeVQVAE	0.029±0.006	1.039±0.072
DiffusionTS	0.020±0.005	0.432±0.048
TOTEM	0.029±0.007	0.724±0.049
TimeTok	**0.002**±0.001	**0.103**±0.014

control. Since these methods cannot inherently generate time series at a specified granularity, we train them on datasets containing paired coarse- and fine-grained representations, allowing conditional generation from a coarse input to a specified target granularity. For unconditional generation, we replace the coarse input with a zero or Null embedding. See details in Appendix A.7.

Conditional Generation. We evaluate conditional generation using two metrics (details in Appendix A). First, we generate five time series for each coarse input, and measure the CRPS_{sum} (lower is better) to evaluate the diversity and its preservation of global structure. TimeTok consistently outperforms all baselines in both datasets. Second, we measure coarse-level consistency (C-Cons) by re-coarsening the generated time series and computing the RMSE against the original coarse input. TimeTok again achieves the best performance across all baselines.

Unconditional Generation. Using the [BOS] token, TimeTok generates time series at granularity levels 1 through 8, respectively; level 8 corresponds to standard generation. Table 4 reports FID averaged over all levels, showing that TimeTok achieves the best performance compared to baselines.

Token-Space Refinement Matters. To test whether TimeTok's gains come from the coarse-to-fine factorization alone or from learning it in a structured token space, we compare against a raw-space AR Transformer that directly models the full refinement trajectory (*i.e.,* direct implementation of Eq. (1)). As shown in Table 16, TimeTok consistently improves both distributional accuracy and coarse consistency, indicating that GC-TSG requires not only a coarse-to-fine objective, but a representation in which coarse-to-fine refinement is compact, ordered, and generative.

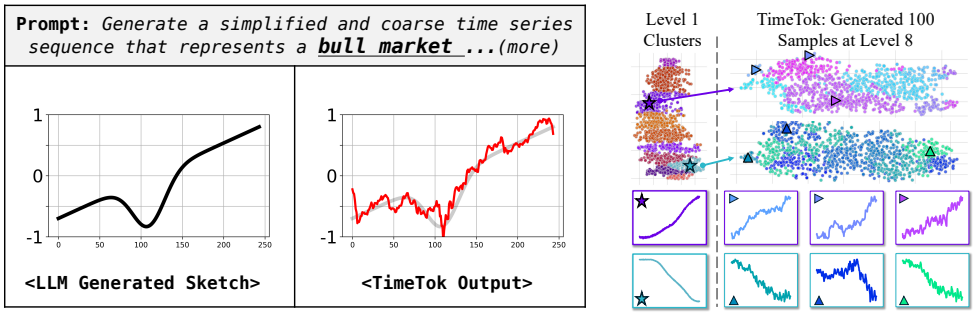
Bridging Temporal Granularity Mismatch. We have performed an additional experiment to verify that coarse-to-fine generated time series are indeed helpful in the downstream tasks (classification), particularly under temporal granularity mismatch between the model's training data and test samples. Specifically, we evaluate classifiers trained on fine-grained ECG signals under four test settings: classifiers tested on (1) coarse time series (obtained by downsampling followed by linear interpolation), (2) coarse time series with Gaussian noise added, (3) refined time series by TimeTok, and (4) original fine-grained time series (Oracle). This setup reflects the realistic scenario discussed in the Introduction, where consumer-grade devices provide only low-resolution ECG signals, whereas hospital-trained models are trained using fine-grained time series. Table 5 demonstrates that TimeTok can effectively adjust temporal granularity without deviating from the underlying data distribution.

Table 5: **Bridging Temporal Granularity Mismatch.** Classification on ECG5000 with different granularity.

ECG5000	ACC ↑
Oracle	$0.959_{\pm 0.001}$
Coarse Time Series	$0.314_{\pm 0.046}$
Coarse Time Series (+Noise)	$0.314_{\pm 0.045}$
Refined Time Series (TimeTok)	$\mathbf{0.918}_{\pm \mathbf{0.006}}$

Table 6: **Foundation tokenizer evaluation** on ECG5000 and Nasdaq datasets. Metrics: FID (lower is better), ACC, F1, and MSE.

	ECG5000			Nasdaq	
Method	FID ↓	ACC ↑	F1 ↑	FID ↓	MSE ↓
TimeVQVAE	0.367	0.924	0.450	0.082	0.047
TOTEM	0.035	0.929	0.394	0.068	0.045
TimeTok **(w/o DFA)**	0.042	0.944	0.583	0.111	0.044
TimeTok **(w/ DFA)**	**0.010**	**0.951**	**0.668**	**0.046**	**0.042**



(A) Generation From Coarse Sketch **(B) Cluster Analysis**

Figure 4: **(A) Generation From Coarse Sketch.** We visualize fine-grained time series generated from coarse sketches provided by the user. **(B) Cluster Analysis.** Using the token in Level 1 as a condition, we generate samples from Level 8. We observe high diversity, showcasing TimeTok's performance as a generative model.

5.3 Extension to Foundational Time-series Tokenization

In the foundational setting, we train TimeTok on UTSD [25]. We apply the same $\mathcal{R}_\ell$'s used in the individual setup, but determine the absolute granularity level for each time series using the DFA scaling exponent, which is discretized into levels via uniform binning. Based on these estimated granularities, we first train the hierarchical tokenizer on UTSD and then train the VAR transformer on downstream tasks using this foundational tokenizer. Details are provided in Appendix A.4

In Table 6, we evaluate the standard generation performance of our foundational tokenizer using two variants of TimeTok: (i) w/o DFA, using the individual setup, and (ii) w/ DFA, using the DFA-based granularity levels. For fair comparison, we also train TimeVQVAE and TOTEM on UTSD. Both variants outperform TimeVQVAE, with the DFA-based variant showing a substantial performance gain. We attribute this improvement to training on a large, heterogeneous dataset with diverse temporal granularities, which helps the tokenizer better capture granularity-specific structures. These results underscore TimeTok's promise as a foundational tokenizer for time series.

5.4 Qualitative and Additional Analysis

User-provided Coarse Sketch. We generate a coarse input via an LLM (*e.g., an increasing stock trend* in Figure 4-A) and use TimeTok to transform this coarse sketch into a fine-grained, realistic time series. See full results in Appendix H.

Clustering Analysis by Granularity. We perform a clustering analysis of the generated time series based on their granularity level (Figure 4-B). As illustrated in the figure, the model generates diverse samples enriched with fine-grained details, while preserving the coarse trend in Level 1.

Token-usage Analysis. We analyze the token-usage pattern of TimeTok on Nasdaq in Appendix J.

GC-TSG. We visualize the coarse-to-fine refinement process in GC-TSG (Appendix F).

Information-Reducing Operators. We evaluate the robustness of our method using a test set constructed with various information-reducing operators (Appendix G).

Ablation. We conduct further ablation studies on key design components of TimeTok to assess their impact and explore alternative design choices (Appendix D.3).

6 Conclusion

In this work, we proposed `TimeTok`, a novel time-series generation model capable of generating time series at any target granularities, whether conditioned on coarse-grained inputs or generated entirely from scratch. This flexibility is made possible with our proposed hierarchical tokenization strategy, where simply controlling the number of tokens for decoding controls the granularity level of the output – enabling truly user-driven generation of time series. We envision `TimeTok` as a promising generative model that can enable the acquisition of diverse multi-granularity data needed for training time-series foundation models, while also solving a wide range of practical, real-world problems.

Acknowledgements We thank the NeurIPS reviewers for their valuable comments and suggestions. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korean government (MSIT) (No. RS-2024-00358602); the Institute of Information & Communications Technology Planning & Evaluation (IITP) grants funded by the Korean government (MSIT), the Artificial Intelligence Graduate School Program (No. RS-2019-II190079, Korea University) and the AI Research Hub Project (No. RS-2024-00457882); and the National Institute for Mathematical Sciences (NIMS) grant funded by the Korean government (MSIT) (No. S26120000). There are no competing interests to declare.

References

[1] Nino Isakadze and Seth S Martin. How useful is the smartwatch ecg? *Trends in cardiovascular medicine*, 30(7):442–448, 2020. 2

[2] Fernando Moreno-Pino and Stefan Zohren. Deepvol: Volatility forecasting from high-frequency data with dilated causal convolutions. *Quantitative Finance*, 24(8):1105–1127, 2024. 2

[3] Yoontae Hwang, Youngbin Lee, Junhyeong Lee, Stefan Zohren, Jang Ho Kim, Woo Chang Kim, Yongjae Lee, and Frank J Fabozzi. Deep learning in asset management: Architectures, applications, and challenges. *Applications, and Challenges (September 15, 2025)*, 2025. 2

[4] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. Time-series generative adversarial networks. *Advances in neural information processing systems*, 32, 2019. 3, 7, 20, 22

[5] Lifeng Shen, Xuyang Li, and Lele Long. Tsgdiff: Rethinking synthetic time series generation from a pure graph perspective. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 25304–25311, 2026. 3, 7, 20, 22

[6] Daesoo Lee, Sara Malacarne, and Erlend Aune. Vector quantized time series generation with a bidirectional prior model, 2023. 3, 7, 20, 22

[7] Xinyu Yuan and Yan Qiao. Diffusion-ts: Interpretable diffusion for general time series generation. In *The Twelfth International Conference on Learning Representations*, 2024. 3, 7, 20, 22

[8] Ilan Naiman, Nimrod Berman, Itai Pemper, Idan Arbiv, Gal Fadlon, and Omri Azencot. Utilizing image transforms and diffusion models for generative modeling of short and long time series. *Advances in Neural Information Processing Systems*, 37:121699–121730, 2024. 3, 7, 20, 22

[9] Sabera J Talukder, Yisong Yue, and Georgia Gkioxari. Totem: Tokenized time series embeddings for general time series analysis. *Transactions on Machine Learning Research*, 2024. 3, 7, 20, 22

[10] Luca Masserano, Abdul Fatir Ansari, Boran Han, Xiyuan Zhang, Christos Faloutsos, Michael W Mahoney, Andrew Gordon Wilson, Youngsuk Park, Syama Sundar Rangapuram, Danielle C Maddix, et al. Enhancing foundation models for time series forecasting via wavelet-based tokenization. In *Forty-second International Conference on Machine Learning*, 2025. 3, 20, 22

[11] Jaeho Kim and Seulki Lee. Transpl: Vq-code transition matrices for pseudo-labeling of time series unsupervised domain adaptation. In *Forty-second International Conference on Machine Learning*, 2025. 3, 20, 22

[12] Keyu Tian, Yi Jiang, Zehuan Yuan, BINGYUE PENG, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. 3, 6, 20, 22

[13] Sucheng Ren, Qihang Yu, Ju He, Xiaohui Shen, Alan Yuille, and Liang-Chieh Chen. Flowar: Scale-wise autoregressive image generation meets flow matching. In *Forty-second International Conference on Machine Learning*. 3, 20, 22

[14] Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. Flextok: Resampling images into 1d token sequences of flexible length. In *Forty-second International Conference on Machine Learning*, 2025. 3, 20, 22

[15] Shan Zhao and Guo-Wei Wei. Jump process for the trend estimation of time series. *Computational Statistics & Data Analysis*, 42(1-2):219–241, 2003. 4

[16] C-K Peng, Shlomo Havlin, H Eugene Stanley, and Ary L Goldberger. Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series. *Chaos: an interdisciplinary journal of nonlinear science*, 5(1):82–87, 1995. 4, 15

[17] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017. 5

[18] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023. 5

[19] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *The Twelfth International Conference on Learning Representations*, 2024. 5

[20] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. 5

[21] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. 6

[22] Xinyao Fan, Yueying Wu, Chang Xu, Yuhao Huang, Weiqing Liu, and Jiang Bian. Mg-tsd: Multi-granularity time series diffusion models with guided learning process. In *The Twelfth International Conference on Learning Representations*, 2024. 7, 16

[23] Ilya Sutskever, James Martens, and Geoffrey E Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 1017–1024, 2011. 7

[24] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016. 7

[25] Yong Liu, Haoran Zhang, Chenyu Li, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. Timer: Generative pre-trained transformers are large time series models. *arXiv preprint arXiv:2402.02368*, 2024. 7, 9

[26] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F Schmidt, Jonathan Weber, Geoffrey I Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 34(6):1936–1962, 2020. 7

[27] Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pages 11121–11128, 2023. 7

[28] Hoang Anh Dau, Anthony Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. 14

[29] Zhen Liu, Yicheng Luo, Boyuan Li, Emadeldeen Eldele, Min Wu, and Qianli Ma. Learning soft sparse shapes for efficient time-series classification. *arXiv preprint arXiv:2505.06892*, 2025. 14

[30] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. 15

[31] Xinyao Fan, Yueying Wu, Chang Xu, Yuhao Huang, Weiqing Liu, and Jiang Bian. Mg-tsd: Multi-granularity time series diffusion models with guided learning process. *arXiv preprint arXiv:2403.05751*, 2024. 17

[32] Yang Li, Han Meng, Zhenyu Bi, Ingolv T Urnes, and Haipeng Chen. Population aware diffusion for time series generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 18520–18529, 2025. 17

[33] Fabrizio Garuti, Enver Sangineto, Simone Luetto, Lorenzo Forni, and Rita Cucchiara. Diffusion transformers for tabular data time series generation. *arXiv preprint arXiv:2504.07566*, 2025. 17

[34] Tal Gonen, Itai Pemper, Ilan Naiman, Nimrod Berman, and Omri Azencot. Time series generation under data scarcity: A unified generative modeling approach. *arXiv preprint arXiv:2505.20446*, 2025. 17

[35] Yu-Hao Huang, Chang Xu, Yueying Wu, Wu-Jun Li, and Jiang Bian. Timedp: Learning to generate multi-domain time series with domain prompts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 17520–17527, 2025. 17

[36] Christopher M Bishop. Mixture density networks. 1994. 17

[37] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022. 18

[38] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024. 20

[39] Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, et al. Chronos-2: From univariate to universal forecasting. *arXiv preprint arXiv:2510.15821*, 2025. 20

[40] Jufang Duan, Shenglong Xiao, and Yuren Zhang. Srt: Super-resolution for time series via disentangled rectified flow. *arXiv preprint arXiv:2606.07605*, 2026. 21

[41] Ivan Skorokhodov, Sharath Girish, Benran Hu, Willi Menapace, Yanyu Li, Rameen Abdal, Sergey Tulyakov, and Aliaksandr Siarohin. Improving the diffusability of autoencoders. *arXiv preprint arXiv:2502.14831*, 2025. 21

[42] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015. 33

Appendix

Table of Contents

A EXPERIMENT DETAILS **14**
- A.1 Gaussian kernel smoothing 14
- A.2 DATASETS 14
- A.3 Level Measurement 15
- A.4 Extension to Foundational Tokenizer 15
- A.5 Evaluation Metric 16
- A.6 Baseline Selection for GC-TSG 17
- A.7 Baseline Implementation for GC-TSG 17

B Limitations and Broader Impacts **19**

C Related Works **20**
- C.1 Time-Series Generation 20
- C.2 Multi-Granular and Discrete Time-Series Representation Learning 20
- C.3 Multi-Scale Token Generation in Computer Vision 20
- C.4 Difference with Temporal Super-Resolution 21
- C.5 Relation to Equivariance Learning 21

D ADDITIONAL EXPERIMENTAL RESULTS **23**
- D.1 Compute Resource and Experiment Environment 23
- D.2 Full Results 23
- D.3 Ablation Study 25
- D.4 Granularity Monotonicity and Smoothing Supervision 26
- D.5 Long-Sequence Generation and Model Size 27
- D.6 Model Parameter Count 27

E Coarse To Fine Reconstruction **28**

F GC-TSG Generation **30**

G Other Information Reducing Operations **31**

H Use Case Analysis **32**

I The Use of LLMs in this work **32**

J Token Usage Analysis on the Nasdaq Dataset **33**

K Robustness Between Evaluators **34**

A EXPERIMENT DETAILS

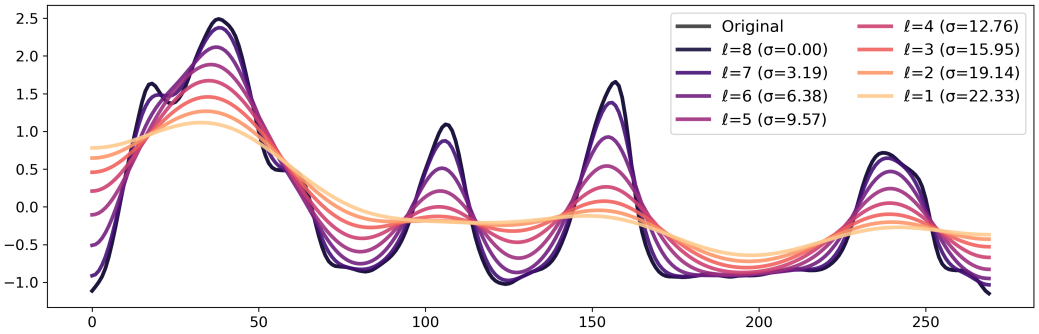


Figure 5: Example of Gaussian kernel smoothing with varying ℓ values. The original signal (*Original*) is shown together with smoothed signals for different ℓ. Note that $\ell = 8$ is identical to the original signal.

A.1 Gaussian kernel smoothing

Target signal $\mathbf{x}^{(\ell)}$ is constructed by smoothing the original sequence $\mathbf{x}^{(L)} \stackrel{\text{def}}{=} \mathbf{x}$ with a Gaussian kernel, $\mathbf{x}^{(\ell)} = g_{\sigma(\ell)} * \mathbf{x}$, where $g_{\sigma(\ell)}$ denotes a discrete normalized Gaussian kernel, *i.e.,* $\sum_i g_\sigma[i] = 1$. Specifically, the kernel weights are given by $g_{\sigma(\ell)}[i] \propto \exp\left(-\frac{i^2}{2\cdot\sigma(\ell)^2}\right)$ and truncated to the window $|i| \leq 3 \cdot \sigma(\ell)$. The bandwidth $\sigma(\ell)$ is defined as a function of the level ℓ:

$$\sigma(\ell) = \sigma_{\max} \cdot \left(\frac{L-\ell}{L-1}\right)$$

where we set $\sigma_{\max} = T/12$ in practice, which results in an effective maximum kernel support of approximately $6 \cdot \sigma_{\max} = T/2$ due to truncation at $|i| \leq 3 \cdot \sigma(\ell)$. This design assigns stronger smoothing to smaller ℓ, yielding coarse representations dominated by long-range structure, while gradually reducing the smoothing as ℓ increases so that higher levels progressively recover fine-grained details. Since Gaussian kernel smoothing is not uniquely invertible, recovering fine-scale details requires learning progressive refinement across higher granularity levels.

A.2 DATASETS

Table 7: **Dataset Statistics.** For each dataset, we report the number of samples in the original train/test splits, the total number of samples, the number of classes, and the length of the time series. We use the first channel of ETTh1. For the UTSD, we report the actual number of samples used to train `TimeTok`.

ID	Dataset Name	# Train	# Test	# Total	# Classes	Input Len	Pred Len	TS Len
1	ECG5000	500	4500	5000	5	–	–	140
2	ItalyPowerDemand	67	1029	1096	2	–	–	24
3	Nasdaq	5320	1520	6840	–	196	48	244
4	ETTh1	4900	1400	6300	–	96	48	144
5	UTSD	2,954,239	-	2,954,239	–	-	-	256

The ECG5000 and ItalyPowerDemand are datasets sourced from the UCR Time Series Classification Archive [28]. For the UCR datasets, we follow the train-validation-test split of 60%-20%-20% as in [29]. ETTh1 is sourced from https://github.com/thuml/autoformer. UTSD can be accessed from https://huggingface.co/datasets/thuml/UTSD. Nasdaq dataset will be made publicly available.

A.3 Level Measurement

Given a time series with unknown granularity, `TimeTok` must identify the appropriate granularity level to perform GC-TSG. While users can manually specify the level based on prior knowledge, such information may not always be available. To address this, we implement an automatic level measurement based on the bias-variance tradeoff principle [30]:

① **Time-Series Tokenization.** We tokenize the input time series at the finest granularity level, retaining all tokens.

② **Distance Measurement.** We decode the tokens at each granularity level $\ell \in \{1, \ldots, 8\}$ obtaining time-series at each granularity level. We then compute the L2 distance between the input time series and each reconstructed time series.

③ **Complexity-Simplicity Tradeoff.** Our objective is to balance token efficiency with reconstruction distance. Rather than simply selecting the level with the minimum distance, we identify the level that yields the highest relative improvement (*i.e.,* the steepest decrease in distance relative to the previous level). This approach selects a level that captures the coarse information while avoiding unnecessary complexity.

The use of our level-measurement function was a design choice, where other measurement functions (*e.g.,* Knee-detection) can be used for level detection.

Detection accuracy and impact on generation. On ECG5000, our level detector achieves an exact-match accuracy of 47.5% and an accuracy of 73.4% within ± 1 level. Errors are concentrated at coarse levels, where neighboring granularities are similar: the mean squared difference between adjacent-level signals increases from 0.0038 (levels 1–2) to 0.0472 (levels 7–8). Despite imperfect level detection, its impact on generation is limited in this evaluation. Using detected levels yields a $\mathrm{CRPS}_{\mathrm{sum}}$ of 1.587, compared with 1.509 using ground-truth levels, and both outperform TimeGAN (1.737). These results suggest that confusion between similar neighboring levels has a relatively small effect on conditional generation.

Method	**CRPS$_{\mathrm{sum}}$** ↓
TimeGAN (Best Baseline)	1.737±0.737
`TimeTok` with Level Detect (Ours)	1.587±0.867
`TimeTok` with GT-Level	**1.509±0.883**

Table 8: **Comparison Between Level Measurements.** To further evaluate the sensitivity of our method to the granularity level measurement performed by the proposed measurement module, we additionally conduct the GC-TSG generation task on the ECG5000 dataset using the ground-truth (GT) granularity level. TimeTok with our detection (1.587) performs very close to the oracle setting with GT levels (1.509), with a gap of only 0.078. Both settings substantially outperform the best baseline (TimeGAN, 1.737), confirming that our level measurement module is robust and introduces negligible performance degradation.

A.4 Extension to Foundational Tokenizer

For the sake of developing a simple and accessible methodology in the single-dataset setting, we have so far assumed that the granularity level of a given time-series dataset corresponds to the finest granularity level (*i.e.,* $\mathbf{x}^{(L)} \stackrel{\text{def}}{=} \mathbf{x}$). This assumption, however, may not hold in heterogeneous settings, such as foundation model training, where data naturally spans multiple temporal granularities, and uniformly treating all time series as finest-granular may be inappropriate. To accommodate such heterogeneous settings, we further extend our training methodology to enable learning from time-series data observed at different temporal granularities.

The key difference is that, since the identity $\mathbf{x}^{(L)} \stackrel{\text{def}}{=} \mathbf{x}$ no longer holds, the granularity levels can no longer be defined solely through the information-reducing operators $\mathcal{R}_\ell$. Accordingly, instead of assigning levels based on the strength of the information-reducing operators, we use detrended fluctuation analysis (DFA) exponent [16] as an absolute quantitative measure of temporal fluctuation to determine the granularity level of each observed time series. Specifically, we define the granularity levels according to the following procedure.

① **Constructing multi-granular representations.** For each observed time series $\mathbf{x}$, we apply the information-reducing operators across all levels to construct representations with diverse degrees of temporal granularity.

② **Computing the DFA exponent.** For each resulting representation, we compute the DFA exponent as an absolute quantitative measure of its temporal fluctuation.

③ **Defining level thresholds.** To determine the DFA exponent range associated with each granularity level, we partition the resulting DFA exponents into L uniform bins. Each candidate representation is then assigned to the granularity level corresponding to the bin that contains its DFA exponent.

During training, each multi-granular representation constructed in Step ① is used as input. Given its assigned granularity level, the decoder is conditioned only on the corresponding number of leading tokens and trained to reconstruct the input at that level.

Under this DFA-based training procedure, the granularity level of an input time series is determined by the bin containing its DFA exponent, without relying on the reconstruction-based level measurement in Appendix A.3. The experiments in Table 6 show that the proposed extension to a foundational tokenizer effectively improves performance in heterogeneous settings, by leveraging DFA-based granularity assignment (*i.e.,* w/ DFA).

A.5 Evaluation Metric

CRPS: For conditional GC-TSG, our goal is to evaluate how accurately the model captures the conditional distribution $p(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)})$, where $\mathbf{x}^{(i)}$ denotes coarse time series and $\mathbf{x}^{(j)}$ the corresponding target sequence. Given a condition $\mathbf{x}^{(i)}$, the model generates K samples $\{\hat{\mathbf{x}}^{(j,k)}\}_{k=1}^{K}$ (with K=5 in all experiments) intended to represent the conditional predictive distribution of $\mathbf{x}^{(j)}$. We evaluate the conditional predictive distribution pointwise in time using the Continuous Ranked Probability Score (CRPS), a strictly proper scoring rule for univariate predictive distributions.

At each time step t, CRPS is estimated from the generated samples $\{\hat{x}_t^{(j,k)}\}_{k=1}^{K}$ and the ground-truth target value $x_t^{(j)}$ as

$$\widehat{\mathrm{CRPS}}(F_t, x_t^{(j)}) = \frac{1}{K}\sum_{k=1}^{K}|\hat{x}_t^{(j,k)} - x_t^{(j)}| - \frac{1}{2K^2}\sum_{k=1}^{K}\sum_{k'=1}^{K}|\hat{x}_t^{(j,k)} - \hat{x}_t^{(j,k')}|.$$

To assess the quality of the entire conditional trajectory, we aggregate the pointwise scores over the temporal horizon and define

$$\mathrm{CRPS}_{\mathrm{sum}} = \sum_{t=1}^{T}\widehat{\mathrm{CRPS}}(F_t, x_t^{(j)}).$$

Following prior work on time-series evaluation, we report $\mathrm{CRPS}_{\mathrm{sum}}$ averaged over the test set [22]. A lower $\mathrm{CRPS}_{\mathrm{sum}}$ indicates that the generated samples more accurately represent the conditional distribution $p(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)})$, both in terms of accuracy and uncertainty calibration.

Coarse consistency: In addition to CRPS, we measure *coarse consistency*, which evaluates whether generated target sequences remain consistent with the provided coarse condition. Recall that information-reducing operator is defined as $\mathbf{x}^{(i)} = \mathcal{R}_i(\mathbf{x}^{(L)})$, *i.e.,* $\mathcal{R}_i$ is explicitly specified as a mapping from the finest-grained observations $\mathbf{x}^{(L)}$ to $\mathbf{x}^{(i)}$. Therefore, $\mathcal{R}_i$ is not directly applicable to an arbitrary intermediate-level signal $\mathbf{x}^{(j)}$. To define a consistency metric for a generated sample $\hat{\mathbf{x}}^{(j)}$, we instead require an operator that maps level j to level i (with $i < j$), which we denote by $\mathcal{R}_{j\to i}$. We then recoarsify each generated target sample via $\mathcal{R}_{j\to i}(\hat{\mathbf{x}}^{(j)})$ and compute the RMSE against the ground-truth condition $\mathbf{x}^{(i)}$:

$$\text{C-cons} = \mathrm{RMSE}\Big(\mathcal{R}_{j\to i}(\hat{\mathbf{x}}^{(j)}), \mathbf{x}^{(i)}\Big),$$

averaged over generated samples and test pairs. This metric is well-defined when $\mathcal{R}$ is instantiated as Gaussian smoothing. In this case, a level is parameterized by the Gaussian standard deviation, and Gaussian filters are closed under composition: applying Gaussians with σ_1 and σ_2 sequentially is equivalent to a single Gaussian with $\sqrt{\sigma_1^2 + \sigma_2^2}$. Hence, given the known smoothing parameters for levels i and j, we can construct $\mathcal{R}_{j\to i}$ by applying an additional Gaussian whose variance matches

the difference, *i.e.,* $\sigma^2_{j\to i} = \sigma^2_i - \sigma^2_j$. For other information-reducing operators, an unambiguous mapping $\mathcal{R}_{j\to i}$ is generally unavailable, so C-cons is reported only for Gaussian smoothing and omitted otherwise.

A.6 Baseline Selection for GC-TSG

In selecting baselines, we applied the following criteria to ensure a fair and reproducible comparison:

① Official code is publicly available.
② The method addresses **unconditional** (and preferably also **class-conditional**) time-series generation, excluding methods developed for classification or forecasting [31].
③ Applicable to **univariate** time series, i.e., not methods developed specifically for **multivariate** setting [32].
④ Designed for **general time-series generation**, rather than only for a specialized format (e.g., tabular-specific settings [33]).
⑤ Not restricted to a highly specific scenario such as **data-scarcity-oriented** [34] or **multi-domain generation** [35].

Critically, a fair baseline must be applicable to **GC-TSG** without introducing unfair architectural advantages. Recent methods such as ImagenTime and TSGDiff are strong performers on standard generation, but applying them to GC-TSG requires **substantial architectural modifications.** That is, **ImagenTime** operates in a Fourier-domain where controlling temporal granularity levels would itself make an entirely new line of research, while **TSGDiff** further converts the Fourier-domain representation into a graph-structured form, adding another layer of complexity that makes granularity-controllable generation fundamentally incompatible with its core design. Such modifications go far beyond minor adaptations and can potentially make the comparison less rigorous. In contrast, ToTem and the other selected baselines could be extended to GC-TSG under a reasonable adaptation protocol, as described in A.7. For TimeGAN and TSGDiff, which do not natively support class-conditional generation, we train separate models for individual classes.

A.7 Baseline Implementation for GC-TSG

Directly realizing Eq. (1) in the raw time-series space is fundamentally challenging. Each refinement step – *i.e.,* $p_\theta(\mathbf{x}^{(\ell)}|\mathbf{x}^{(<\ell)})$ – requires the model to condition on the full trajectory of all preceding coarser time-series representations, which makes training computationally expensive and often unstable for long time series. Common workarounds – such as modeling $p_\theta(\mathbf{x}^{(j)}|\mathbf{x}^{(i)})$ for all granularity pairs – fail to impose an explicit hierarchical ordering over temporal granularities, which may struggle to enforce consistent coarse-to-fine refinement or to capture dependencies across distant temporal granularities.

To empirically substantiate these considerations, we present a comparative study of baseline implementations that realize GC-TSG in different representational spaces. Among them, the **AR Transformer** directly realizes Eq. (1) in the raw time-series space by modeling the full coarse-to-fine refinement trajectory, whereas the remaining baselines approximate GC-TSG by modeling the pairwise conditional $p_\theta(\mathbf{x}^{(j)} \mid \mathbf{x}^{(i)})$ across granularity pairs. Specifically, we examine three variants that differ in where the hierarchical refinement is instantiated. A detailed comparison with the AR Transformer is provided in Table 16 for conditional generation and Table 17 for unconditional (marginal) generation.

- **GC-TSG in raw time-series space.**
  RNN-TF/PF. We implement RNN-TF and RNN-PF baselines using an encoder-decoder architecture with Gated Recurrent Units (GRUs) and a Mixture Density Network (MDN) [36] head. The models map the target granularity level $\ell \in \{1, \ldots, 8\}$ to a learnable embedding vector $\mathbf{e}_\ell$, which is injected into the decoder as a global condition. To incorporate the coarse condition $\mathbf{c}$, the encoder summarizes the sequence to initialize the decoder's hidden state. Additionally, at each time step t, the decoder receives a concatenated input vector consisting of the previous output x_{t-1}, the level embedding $\mathbf{e}_\ell$, and the local condition value c_t. The MDN head maps the decoder's hidden states to the parameters of a Gaussian Mixture Model for probabilistic generation. **RNN-TF** is trained via teacher forcing to minimize the negative log-likelihood, while **RNN-PF** employs adversarial training with a Bi-directional GRU

discriminator to align hidden state dynamics and mitigate exposure bias. For marginal generation, $\mathbf{c}$ is replaced with a zero-valued sequence.

Diffusion-TS. We extend the Diffusion-TS by incorporating the target granularity level and a coarse-condition. The model encodes the target granularity level $\ell \in \{1, \ldots, 8\}$ into a vector embedding $\mathbf{e}_\ell$ using an MLP, which is injected into the Transformer backbone via Adaptive Layer Normalization alongside timestep embeddings. To incorporate the coarse condition $\mathbf{c}$, we concatenate it with the noisy input along the feature dimension. The model is trained to reconstruct the target time series at level ℓ from the noisy input, conditioned on $\mathbf{e}_\ell$ and $\mathbf{c}$. For marginal generation, conditioned on $\mathbf{e}_\ell$ alone, we replace $\mathbf{c}$ with a learnable `Null` embedding and train to reconstruct the target time series at level ℓ.

AR Transformer. We train an autoregressive Transformer with the MDN head on the concatenated 8-level sequence $\mathbf{s} = \mathbf{x}^{(1)} \oplus \ldots \oplus \mathbf{x}^{(8)} \in \mathbb{R}^{8T}$. The model uses learnable positional embeddings. To indicate level transitions, we insert a learnable [SEP] token at each level boundary with corresponding level embedding $\mathbf{e}_\ell$ added to [SEP] (and [BOS] for level 1). For unconditional generation, we begin from [BOS] and generate sequentially up to the desired target level. For conditional generation, given a condition at level i, the model requires the full trajectory up to that level i (i.e., $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(i)}$) as context. Unlike `TimeTok`, where coarser levels are naturally obtained during tokenization, the AR Transformer cannot derive them from a single-level condition. Therefore, we provide the ground-truth trajectory $\{\mathbf{x}^{(\ell)}\}_{\ell=1}^{i}$ as a prefix and generate levels $i{+}1$ through the target level autoregressively.

- **GC-TSG in continuous latent space.**

  TimeGAN. We extend the TimeGAN framework to support granularity-controllable generation. The generator receives $\boldsymbol{\epsilon} \oplus \mathbf{c} \oplus \mathbf{e}_\ell$ as input, where $\boldsymbol{\epsilon}$ is random noise, $\mathbf{c}$ is the coarse condition sequence, and $\mathbf{e}_\ell$ is the target level embedding. Unlike standard GANs, the generator produces a sequence of embeddings in the latent space, which is then mapped to the time-series space via a recovery network. The discriminator also operates in the latent space, distinguishing real latent representations $\mathbf{h}$ (derived from real time series via an embedding network) from generated ones $\hat{\mathbf{h}}$ using the concatenated input $\mathbf{h} \oplus \mathbf{c} \oplus \mathbf{e}_\ell$. To enable marginal generation, we omit the coarse condition (*i.e.,* $\mathbf{c} = \mathbf{0}$) for unconditional training samples, allowing the model to generate samples conditioned solely on the target granularity level.

- **GC-TSG in discrete token space.**

  TimeVQVAE. We adapt the two-stage TimeVQVAE framework for granularity-controllable generation. In Stage 1, we train the VQ-VAE tokenizer on time series at all granularity levels $\ell \in \{1, \ldots, 8\}$ by randomly sampling $\mathbf{x}^{(\ell)}$ during training, enabling the tokenizer to learn a unified discrete representation across varying granularities. In Stage 2, we extend the MaskGIT [37] by concatenating condition tokens, a [SEP] token, and masked target tokens. To encode the target granularity level, we add a learnable level embedding $\mathbf{e}_\ell$ to the [SEP] token. For marginal generation, we replace the condition tokens with a learnable `Null` embedding and generate conditioned on [SEP]$+\mathbf{e}_\ell$ alone. For conditional generation, we tokenize the condition time series and generate conditioned on both the condition prefix and [SEP]$+\mathbf{e}_\ell$.

  TOTEM. We adapt the two-stage TOTEM framework for granularity-controllable generation. In Stage 1, we train a VQ-VAE tokenizer that directly processes raw time series across all granularity levels $\ell \in \{1, \ldots, 8\}$, enabling the model to learn a unified discrete representation for both high-resolution targets and coarsified signals. In Stage 2, we implement an Autoregressive Transformer to handle the generative process. The transformer receives a prefix consisting of the tokens from the source condition time series and a [SEP] token, where the target level information is integrated into the [SEP] token via a learnable level embedding $\mathbf{e}_\ell$. During training, the model is optimized using Teacher Forcing to predict the subsequent target tokens sequentially in an autoregressive manner. For marginal generation, the source condition sequence is replaced by a zero-valued input, and the model generates the target time series based on the [SEP] token and its associated level information alone.

B Limitations and Broader Impacts

Limitations. This work introduces GC-TSG as a flexible generation task that extends conventional time-series generative modeling by explicitly controlling both input and output temporal granularity. To establish a clear and controlled formulation, our main experiments focus primarily on the single-dataset setting, where granularity levels can be defined through a clear hierarchy of information-reducing operators. Although we further provide an extension of `TimeTok` to heterogeneous multi-domain data in Section 5.3 and Appendix A.4, our evaluation of foundation-scale tokenization remains preliminary. A more comprehensive study involving broader downstream tasks, additional domains, and practical use cases is an important direction for future work.

Our experiments target univariate time series, and multivariate extensions remain future work. Each model is trained at a fixed sequence length, so results across different lengths do not establish mixed-length inference within a single model. Furthermore, our cross-domain evaluation concerns tokenizer transferability rather than zero-shot refinement by the full generative model, which remains unverified. Refinements should be interpreted as plausible samples conditioned on coarse observations, rather than recovery of the unknown ground-truth fine-scale trajectory.

Broader Impacts. `TimeTok` has the potential to support a wide range of applications involving time-series data, particularly in domains where practitioners need to access, analyze, or generate data across multiple temporal granularities. Nevertheless, generated samples should undergo expert validation before being used for decision-making or downstream deployment, especially in high-stakes domains such as healthcare and finance.

C Related Works

C.1 Time-Series Generation

Deep generative models have achieved strong performance across modalities, but time-series generation remains challenging because temporal data exhibit long-range dependencies, local fluctuations, nonstationarity, and domain-specific dynamics. TimeGAN [4] adapts generative adversarial networks to time series by using supervised forecasting signals to improve temporal coherence. TimeVQVAE [6] learns discrete time-series representations through vector quantization and performs generation in the resulting latent space. Diffusion-based methods have further improved sample quality through specialized inductive biases, including frequency-domain losses in Diffusion-TS [7], graph-based temporal modeling in TSGDiff [5], and image-based representations in ImagenTime [8].

These methods address the standard generation problem: learning to sample time series at the fixed temporal granularity specified by the training data. They do not natively expose temporal granularity as a controllable variable. In particular, they do not learn a single conditional family $p(x^{(j)} \mid x^{(i)})$ over arbitrary input-output granularity pairs, nor do they provide a mechanism for refining a coarse observation or user sketch into plausible finer-scale trajectories. Extending them to this setting requires adding external conditioning mechanisms, training separate pairwise models, or changing the architecture. TimeTok instead treats granularity control as part of the generative objective itself.

This distinction is important because GC-TSG is not only standard generation with a different sampling rate. It requires a model that can preserve coarse structure while introducing statistically plausible finer dynamics, and that can generate at different target granularities from the same learned representation. Thus, the central question is not only whether generated samples look realistic at one resolution, but whether a model can traverse a learned coarse-to-fine temporal hierarchy while maintaining consistency with the conditioning signal.

C.2 Multi-Granular and Discrete Time-Series Representation Learning

Recent time-series tokenization methods have shown that discrete representations can support scalable and general-purpose temporal modeling. TOTEM [9] learns tokenized embeddings for time-series analysis, while WaveToken [10] uses wavelet decompositions to capture time-localized frequency structure. TransPL [11] decomposes temporal patches into trend and residual components and quantizes them using separate codebooks. These methods introduce useful structure into the latent space, but their tokenization is not designed as a controllable generative interface. The number of tokens is typically fixed or determined by the representation scheme, rather than by the desired output granularity. Consequently, they do not make prefix length semantically correspond to temporal granularity, and they do not couple tokenization with a generative process that can move from coarse to fine levels.

Tokenization is also central to recent time-series foundation models. Chronos [38] quantizes scaled observations into a fixed vocabulary using uniform binning, whereas Chronos-2 [39] embeds patches of continuous values together with time indices and observation masks. Neither scheme explicitly aligns token prefixes with temporal granularity levels. In contrast, `TimeTok` organizes tokens so that successive tokens carry progressively finer temporal granularity, making the token order itself a controllable interface over granularity.

C.3 Multi-Scale Token Generation in Computer Vision

Coarse-to-fine generation has been studied extensively in computer vision. VAR [12] and FlowAR [13] replace token-by-token generation with next-scale generation: coarse visual tokens are generated first and condition subsequent finer-scale tokens. This has proven effective for improving image generation quality and scaling autoregressive image models. FlexTok [14] takes a related but distinct direction by using nested dropout to make shorter token prefixes capture simpler image content, enabling efficient generation with variable token budgets. TimeTok is inspired by this broader idea of ordered token generation, but the problem it solves is different. In image generation, scale usually refers to spatial resolution, visual detail, or token efficiency. The generated object is still typically evaluated as a final full-resolution image, and intermediate scales are mainly a latent scaffold for better generation. In GC-TSG, temporal granularity is itself the user-facing object of control: the

input may be a coarse time series, the output may be requested at any finer level, and intermediate levels correspond to different temporal dynamics rather than merely different visual resolutions.

This distinction is not cosmetic. Temporal granularity changes what aspects of the dynamics are observable: coarse levels preserve trends and long-range behavior, while fine levels recover transients, fluctuations, and local temporal events. Therefore, a time-series model must learn not only a nested representation, but a conditional refinement process over dynamics. Unlike VAR and FlowAR, TimeTok does not use the hierarchy only to improve final-resolution quality; it exposes the hierarchy as an input-output control mechanism. Unlike FlexTok, TimeTok explicitly models transitions between granularity levels through block-wise autoregressive generation, enabling conditional refinement from arbitrary coarse prefixes.

C.4 Difference with Temporal Super-Resolution

Temporal super-resolution reconstructs a high-resolution signal from low-resolution measurements, and prior work typically treats it as a separate problem from time-series generation [40]. TimeTok differs in two respects. First, the target granularity is a user-specified control variable, so a single model refines from any source level to any finer level rather than learning a fixed (low $\rightarrow$ high) mapping. Second, TimeTok also supports unconditional generation at any granularity, of which standard time-series generation is a special case. Super-resolution therefore corresponds to a single conditional instance of the broader GC-TSG problem.

C.5 Relation to Equivariance Learning

Equivariance learning encourages corresponding transformations in the latent and data spaces, for example by training downsampled latents to reconstruct correspondingly downsampled images [41]. Our hierarchical decoding objective similarly aligns token prefixes with corresponding temporal granularities. The key distinction is that TimeTok additionally learns generative transitions across granularities. While the aligned latent-to-data mapping does not itself generate finer latents from coarser ones, TimeTok's block-wise autoregressive model generates missing finer-level tokens conditioned on a coarse prefix. This enables generation at a selected granularity and refinement of coarse observations. The alignment therefore serves as a basis for granularity-controllable generation, rather than the end goal.

Table 9: Comparison of `TimeTok` with related lines of work.

Line of work	**Representative methods**	**Generates realistic time series**	**Controls target granularity at inference**	**Conditions on arbitrary coarser input**	**Ordered prefix has temporal meaning**	**Learns coarse-to-fine refinement trajectory**	**Key distinction from `TimeTok`**
Standard time-series generation	TimeGAN [4], Diffusion-TS [7], TSGDiff [5], ImagenTime [8]	Yes	No	No	No	No	Learns samples at the training granularity; does not model $p(x^{(j)} \mid x^{(i)})$ across granularity pairs.
Discrete time-series generation	TimeVQVAE [6]	Yes	No	No	No	No	Uses discrete latents, but token order does not define a controllable temporal hierarchy.
Time-series tokenization / foundation representations	TOTEM [9]	Representation, not primarily generation	No	No	No	No	Learns useful tokens for downstream tasks, but not a generative interface for granularity control.
Multi-granular time-series tokens	WaveToken [10], TransPL [11]	Not the main goal	No	No	Partly	No	Encodes multi-resolution or trend–residual structure, but granularity is fixed by the representation rather than chosen by the user.
Multi-scale image generation	VAR [12], FlowAR [13]	Yes, for images	Intermediate scales, mainly latent	No	Spatial/visual scale	Yes, for image latent scales	Uses coarse-to-fine token generation to improve final image quality; `TimeTok` uses hierarchy as user-facing input-output control.
Flexible image tokenization	FlexTok [14]	Yes, for images	Token budget / image complexity	No	Visual complexity	No explicit temporal dynamics	Variable prefixes control image complexity, not temporal refinement from coarse observations.
Raw-space hierarchical GC-TSG baseline	AR Transformer over $(x^{(1)}, \ldots, x^{(L)})$	Yes	Yes	Yes	N/A	Yes, but in raw space	Directly models the right objective but is inefficient and weaker empirically; motivates `TimeTok`'s token-space hierarchy.
`TimeTok`	**This work**	**Yes**	**Yes**	**Yes**	**Yes**	**Yes**	**A single model for generation from scratch, generation at intermediate granularities, and conditional refinement from coarse inputs.**

D ADDITIONAL EXPERIMENTAL RESULTS

In this section, we present additional experiments that we omitted in the main body of the paper due to limited space.

D.1 Compute Resource and Experiment Environment

All experiments were conducted on NVIDIA A6000 GPU (48GB) or RTX PRO 6000 Blackwell (96GB). We have used Python 3.10 to run all experiments, and will release the installation requirements through GitHub.

D.2 Full Results

Table 10: **Marginal (Unconditional) Generation Performance of ECG5000.** FID is evaluated at each granularity level (L1–L8).

Method	L1 ↓	L2 ↓	L3 ↓	L4 ↓	L5 ↓	L6 ↓	L7 ↓	L8 ↓
RNN(TF)	2.778±1.096	1.393±0.500	0.459±0.129	1.121±0.430	0.643±0.150	0.206±0.064	0.286±0.101	0.188±0.057
RNN(PF)	0.649±0.106	0.485±0.079	0.202±0.036	0.170±0.042	0.118±0.044	0.105±0.026	0.199±0.023	0.082±0.021
TimeGAN	0.139±0.024	0.224±0.036	0.263±0.093	0.477±0.143	0.989±0.186	0.872±0.358	1.474±0.414	1.870±0.296
TimeVQVAE	0.045±0.011	0.031±0.005	0.017±0.002	0.017±0.004	0.031±0.003	0.021±0.004	0.036±0.009	0.034±0.011
DiffusionTS	0.017±0.004	0.018±0.002	0.022±0.006	0.019±0.006	0.025±0.006	0.022±0.008	0.018±0.003	0.023±0.004
TOTEM	0.025±0.005	0.020±0.001	0.024±0.005	0.015±0.004	0.028±0.010	0.022±0.005	0.039±0.009	0.058±0.016
TimeTok	**0.001**±0.000	**0.001**±0.001	**0.001**±0.000	**0.001**±0.000	**0.002**±0.001	**0.003**±0.001	**0.003**±0.001	**0.006**±0.001

Table 11: **Marginal (Unconditional) Generation Performance of Nasdaq.** FID is evaluated at each granularity level (L1–L8).

Method	L1 ↓	L2 ↓	L3 ↓	L4 ↓	L5 ↓	L6 ↓	L7 ↓	L8 ↓
RNN(TF)	8.029±0.629	7.490±0.606	8.406±1.007	8.173±0.654	8.859±0.154	11.34±1.167	11.05±2.312	3.362±0.317
RNN(PF)	4.522±0.380	4.047±0.594	4.496±0.243	3.755±0.292	4.009±0.195	4.209±0.440	4.132±0.278	4.421±0.384
TimeGAN	9.043±0.344	9.956±0.738	9.436±0.563	9.528±0.982	9.386±0.419	9.115±0.546	10.247±0.891	9.833±0.781
TimeVQVAE	1.935±0.157	0.596±0.035	1.009±0.095	1.169±0.037	0.853±0.056	0.686±0.040	1.420±0.117	0.645±0.035
DiffusionTS	0.198±0.019	0.252±0.019	0.322±0.040	0.369±0.028	0.539±0.035	0.440±0.077	0.597±0.103	0.739±0.063
TOTEM	0.748±0.038	0.631±0.058	0.589±0.031	0.722±0.057	0.671±0.057	0.795±0.045	0.580±0.022	1.058±0.085
TimeTok	**0.088**±0.010	**0.100**±0.010	**0.121**±0.027	**0.090**±0.016	**0.106**±0.013	**0.130**±0.014	**0.094**±0.007	**0.091**±0.017

Table 12: **Full Results of GC-TSG.** Full results for granularity-controlled time-series generation (GC-TSG) on the ECG5000, ItalyPowerDemand, Nasdaq, and ETTh1 datasets. Here, i denotes the source level and j the target level, *i.e.,* generating level-j time series from level-i input. We report CRPS_{sum} and coarse-consistency (C-Cons), where C-Cons measures the RMSE between the original coarse input and the generated output after re-coarsening with a Gaussian filter.

	ECG5000		ItalyPowerDemand		Nasdaq		ETTh1	
$i \to j$	CRPS_{sum} ↓	C-Cons ↓	CRPS_{sum} ↓	C-Cons ↓	CRPS_{sum} ↓	C-Cons ↓	CRPS_{sum} ↓	C-Cons ↓
$1 \to 2$	1.079±0.408	0.011±0.004	2.785±0.307	0.020±0.005	20.09±16.54	0.091±0.073	21.44±34.86	0.375±2.135
$1 \to 3$	1.037±0.364	0.009±0.004	2.813±0.256	0.019±0.005	15.18±10.77	0.076±0.061	17.38±33.31	0.304±1.821
$1 \to 4$	1.374±0.568	0.010±0.007	2.644±0.307	0.016±0.007	15.02±12.54	0.081±0.079	17.15±35.46	0.338±2.063
$1 \to 5$	1.718±0.526	0.010±0.006	2.431±0.254	0.013±0.007	14.50±11.52	0.083±0.078	18.18±33.65	0.313±1.892
$1 \to 6$	2.173±0.674	0.011±0.007	2.037±0.262	0.009±0.006	16.48±12.47	0.095±0.088	19.45±30.63	0.297±1.798
$1 \to 7$	2.730±0.848	0.011±0.006	1.602±0.399	0.004±0.004	17.11±12.47	0.096±0.088	20.30±23.63	0.191±1.063
$1 \to 8$	3.042±0.900	0.011±0.007	1.061±0.277	0.002±0.002	17.72±12.63	0.089±0.085	23.25±25.00	0.220±1.191
$2 \to 3$	0.969±0.354	0.010±0.004	2.850±0.241	0.021±0.005	14.55±9.16	0.075±0.056	17.98±35.96	0.354±2.118
$2 \to 4$	1.109±0.369	0.010±0.004	2.670±0.296	0.017±0.006	13.73±9.15	0.081±0.072	18.25±39.89	0.420±2.486
$2 \to 5$	1.485±0.556	0.010±0.006	2.380±0.274	0.013±0.006	14.57±10.87	0.090±0.082	18.86±38.85	0.398±2.416
$2 \to 6$	1.897±0.711	0.011±0.006	2.031±0.273	0.010±0.006	15.29±11.02	0.093±0.084	19.61±33.72	0.345±1.992
$2 \to 7$	2.401±0.924	0.011±0.006	1.607±0.398	0.004±0.004	15.58±11.03	0.088±0.082	20.45±26.13	0.226±1.193
$2 \to 8$	2.693±1.001	0.012±0.007	1.004±0.344	0.002±0.004	17.73±11.26	0.093±0.080	21.36±22.91	0.176±1.060
$3 \to 4$	1.022±0.589	0.011±0.006	2.704±0.286	0.019±0.005	13.94±9.85	0.090±0.078	14.25±29.15	0.233±1.688
$3 \to 5$	1.283±0.639	0.011±0.006	2.435±0.232	0.016±0.008	13.56±9.00	0.087±0.075	14.87±29.45	0.224±1.638
$3 \to 6$	1.637±0.776	0.012±0.007	2.081±0.289	0.012±0.006	13.93±10.06	0.084±0.079	16.16±25.64	0.203±1.354
$3 \to 7$	2.026±0.766	0.012±0.007	1.536±0.368	0.005±0.005	16.46±11.22	0.101±0.086	19.81±26.20	0.218±1.156
$3 \to 8$	2.322±0.874	0.012±0.007	0.864±0.285	0.002±0.002	17.82±12.09	0.102±0.089	22.03±26.28	0.216±1.179
$4 \to 5$	0.988±0.360	0.010±0.005	2.442±0.214	0.018±0.008	13.44±9.77	0.094±0.085	16.00±31.17	0.272±1.541
$4 \to 6$	1.265±0.468	0.011±0.006	2.056±0.267	0.013±0.006	14.38±10.80	0.096±0.087	16.81±28.87	0.248±1.362
$4 \to 7$	1.602±0.591	0.012±0.007	1.513±0.371	0.006±0.004	15.81±12.10	0.105±0.095	20.64±29.00	0.274±1.295
$4 \to 8$	1.858±0.638	0.011±0.006	0.810±0.319	0.002±0.003	17.67±14.28	0.105±0.101	23.75±30.20	0.291±1.363
$5 \to 6$	0.906±0.331	0.009±0.005	2.026±0.279	0.014±0.006	14.78±12.08	0.106±0.098	18.59±34.74	0.352±1.636
$5 \to 7$	1.195±0.396	0.009±0.005	1.504±0.390	0.007±0.004	15.34±12.95	0.104±0.101	19.28±29.45	0.275±1.288
$5 \to 8$	1.526±0.528	0.010±0.006	0.750±0.210	0.002±0.002	18.06±15.97	0.114±0.112	20.56±26.87	0.235±1.231
$6 \to 7$	0.914±0.390	0.007±0.005	1.541±0.407	0.008±0.004	16.34±16.50	0.121±0.123	15.22±25.32	0.229±1.130
$6 \to 8$	1.332±0.502	0.008±0.005	0.705±0.186	0.003±0.004	17.81±16.46	0.120±0.119	18.21±25.78	0.236±1.206
$7 \to 8$	0.835±0.378	0.006±0.005	0.664±0.186	0.004±0.003	17.53±16.12	0.129±0.126	16.70±27.14	0.252±1.262

Table 13: **Baseline Comparison of GC-TSG.** We compare `TimeTok` with TOTEM (strongest baseline) in GC-TSG task on the ItalyPowerDemand and ETTh1 datasets. We report the average (Avg) of all $i \to j$ pairs.

	ItalyPowerDemand		ETTh1	
Avg	CRPS_{sum} ↓	C-Cons ↓	CRPS_{sum} ↓	C-Cons ↓
TOTEM	1.099±0.761	0.074±0.041	29.45±13.14	0.358±0.219
`TimeTok`	**0.501**±0.357	**0.027**±0.019	**12.22**±6.245	**0.104**±0.084

Table 14: **Baseline Comparison of Marginal Generation.** We compare `TimeTok` with TOTEM in marginal time series generation task on the ItalyPowerDemand and ETTh1 datasets across 8 layers (L1–L8). We report the FID score.

Method	Dataset	L1 ↓	L2 ↓	L3 ↓	L4 ↓
TOTEM	ItalyPowerDemand	0.061±0.007	0.050±0.007	0.078±0.011	0.084±0.019
`TimeTok`	ItalyPowerDemand	**0.006**±0.000	**0.009**±0.000	**0.006**±0.000	**0.006**±0.000
TOTEM	ETTh1	0.300±0.027	0.246±0.013	0.537±0.051	0.723±0.049
`TimeTok`	ETTh1	**0.029**±0.002	**0.034**±0.001	**0.066**±0.003	**0.038**±0.003

Method	Dataset	L5 ↓	L6 ↓	L7 ↓	L8 ↓
TOTEM	ItalyPowerDemand	0.062±0.008	0.076±0.010	0.120±0.016	0.140±0.033
`TimeTok`	ItalyPowerDemand	**0.007**±0.000	**0.011**±0.002	**0.009**±0.001	**0.006**±0.000
TOTEM	ETTh1	0.547±0.020	0.519±0.030	0.720±0.024	0.568±0.038
`TimeTok`	ETTh1	**0.087**±0.008	**0.093**±0.007	**0.080**±0.008	**0.105**±0.014

Table 15: **Additional Foundation Tokenizer Evaluation..** Evaluation of the UTSD-pretrained tokenizers on ItalyPowerDemand and ETTh1 under the same protocol as Table 6.

	ItalyPowerDemand		ETTh1	
Method	FID ↓	ACC ↑	FID ↓	MSE ↓
TOTEM	0.0466	0.9545±0.0164	0.0916	0.1098±0.0002
`TimeTok` (w/o DFA)	0.0784	0.9788±0.0026	0.2925	0.1057±0.0005
`TimeTok` **(w/ DFA)**	**0.0388**	**0.9818**±0.0045	**0.0593**	**0.1013**±0.0004

D.3 Ablation Study

Table 16: **Conditional Generation.** Conditional GC-TSG evaluation under two settings: Coarse-to-Fine ($i = 1$, $j = 8$) and All-Pairs average (Avg, $i \to j$, $0 < i < j$). C-Cons denotes coarse-level consistency, measured by RMSE.

	Classification (ECG5000)				Forecasting (Nasdaq)			
	C2F ($1 \to 8$)		C2F-All Pairs ($i \to j$)		C2F ($1 \to 8$)		C2F-All Pairs ($i \to j$)	
Method	**CRPS$_{\text{sum}}$** ↓	**C-Cons** ↓	**CRPS$_{\text{sum}}$** ↓	**C-Cons** ↓	**CRPS$_{\text{sum}}$** ↓	**C-Cons** ↓	**CRPS$_{\text{sum}}$** ↓	**C-Cons** ↓
AR Transformer	4.827±1.768	0.027±0.014	5.804±1.658	0.040±0.018	32.57±25.82	0.160±0.136	22.98±18.95	0.119±0.107
`TimeTok`	**3.042**±0.900	**0.011**±0.007	**1.587**±0.867	**0.010**±0.006	**17.72**±12.63	**0.089**±0.085	**15.87**±12.48	**0.096**±0.090

Table 17: Unconditional (marginal) generation performance across granularity levels (L1–L8), evaluated on ECG5000 and Nasdaq.

Dataset	**Method**	**L1** ↓	**L2** ↓	**L3** ↓	**L4** ↓	**L5** ↓	**L6** ↓	**L7** ↓	**L8** ↓
ECG5000	AR Transformer	1.060±0.358	1.713±0.431	1.420±0.469	2.019±0.693	2.918±0.804	3.213±0.684	3.494±0.736	3.938±0.921
Nasdaq	AR Transformer	14.990±0.730	12.427±0.886	13.305±1.111	13.287±0.868	12.870±0.852	13.309±0.622	13.443±1.722	12.405±1.176

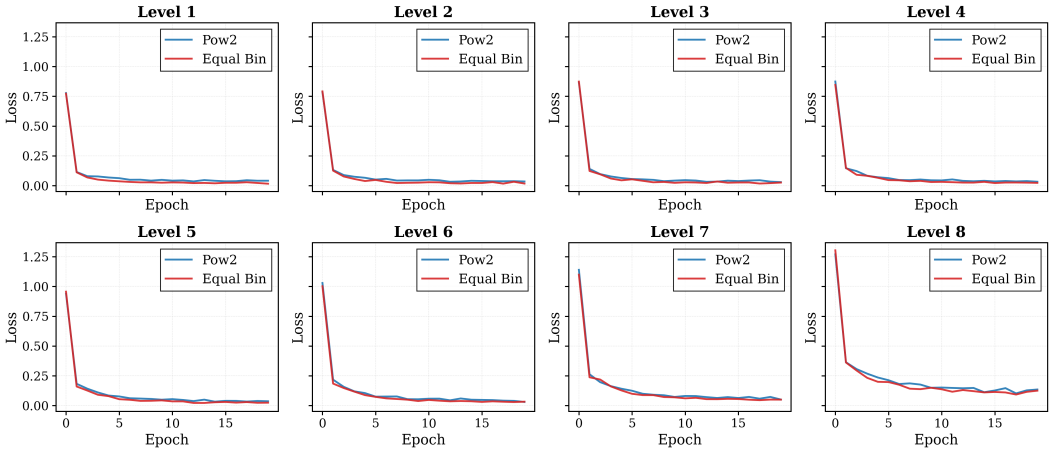


Figure 6: Level-wise training loss comparison between exponential (Pow2) and uniform (Equal Bin) token allocation strategies across all eight granularity levels. Both strategies exhibit nearly identical optimization behavior at every level, indicating that allocating fewer tokens to coarser levels does not adversely affect training dynamics. The x-axis denotes training progress, where one unit corresponds to 10 epochs.

Token Allocation Across Granularity Levels. We study how the conditioning budget affects optimization by comparing our default exponential allocation $n_\ell = 2^{\ell-1}$ against a uniform allocation $n_\ell = 16 \cdot \ell$, while fixing the maximum granularity level to $L = 8$ and the total finest-level budget to $n_L = M = 128$. As shown in Figure 6, the two token allocation strategies exhibit nearly identical training dynamics across all levels. Consistently, validation reconstruction losses on the original time-series data – measured as the mean squared error between generated sequences and the ground

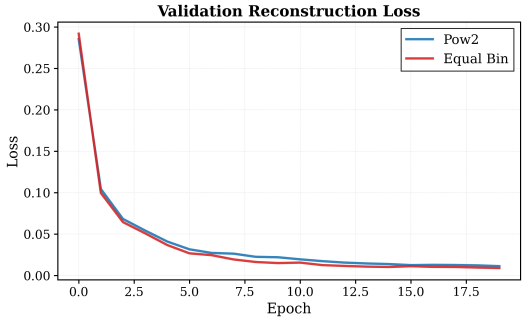


Figure 7: Validation reconstruction loss measured on the original time-series signals, comparing exponential (Pow2) and uniform (Equal Bin) token allocation strategies. The two approaches show nearly identical convergence trends, suggesting comparable reconstruction performance on the validation set.

Table 18: **Long-Sequence Generation.** HandOutlines results (length 2,709). Oracle uses real training data.

Method	FID ↓	ACC ↑	F1 ↑
Oracle	–	0.920±0.019	0.913±0.019
Diffusion-TS	0.621±0.143	0.646±0.003	0.420±0.015
TimeVQVAE	0.041±0.0035	0.675±0.005	0.586±0.022
TimeTok	**0.023**±0.001	**0.760**±0.024	**0.712**±0.055

Table 19: **Model Parameter Counts.** Parameters (M) on Nasdaq and HandOutlines.

Method	Nasdaq	HandOutlines
TimeVQVAE	1.8	16.1
TSGDiff	26.3	30.2
Diffusion-TS	0.6	51.1
ImagenTime	57.6	151.7
TimeTok	9.42	**9.84**

truth –follow similar trajectories under both allocations, as seen in Figure 7. These results suggest that coarse time-series representations require substantially fewer tokens to achieve comparable optimization and reconstruction behavior, making exponential token allocation an efficient design choice under a fixed budget.

Comparison with AR Transformer in Raw Time-Series Space. We compare our approach with an AR Transformer that directly operates in the raw time-series space and explicitly models the full coarse-to-fine refinement trajectory according to Eq. (1). Unlike our method, which approximates refinement via conditional generation across granularity pairs in a structured token space, the AR Transformer must condition on the entire refinement history, leading to unstable learning. The results reveal a clear contrast between conditional and marginal generation behaviors of the AR Transformer. As shown in Table 16, the AR Transformer demonstrates reasonable performance in the conditional setting, where external conditions provide sufficient guidance to partially stabilize the generation process. However, as evidenced in Table 17, its performance degrades substantially in the marginal (unconditional) generation setting. This failure highlights the intrinsic difficulty of learning and sampling the entire coarse-to-fine refinement trajectory in the raw time-series space.

D.4 Granularity Monotonicity and Smoothing Supervision

We evaluate granularity ordering on 274 HandOutlines samples (length 2,709) across eight token budgets $(1, 2, 4, \ldots, 128)$. The mean Spearman correlation between increasing token budgets and decreasing DFA exponents is 0.966. The adjacent-level violation rate, where the DFA exponent increases despite a larger token budget, is 13.5%. Of these violations, 86.9% occur at the one-to-two-token transition; violation rates for transitions from four through 128 tokens are at most 0.7%. When every prefix instead reconstructs the original signal without smoothing supervision, the mean correlation is near zero and the violation rate increases to 47.1%. These results support the importance of smoothing supervision for learning an ordered granularity hierarchy, although strict monotonicity is not guaranteed.

D.5 Long-Sequence Generation and Model Size

We evaluate `TimeTok` on HandOutlines (length 2,709) to examine generation beyond the maximum length of 256 used in our main experiments. As shown in Table 18, `TimeTok` outperforms Diffusion-TS and TimeVQVAE across FID, classification accuracy, and F1, demonstrating its applicability to substantially longer sequences. Each model is trained at the evaluated sequence length; this experiment does not establish mixed-length inference within a single trained model.

D.6 Model Parameter Count

Table 19 compares parameter counts on Nasdaq (length 244) and HandOutlines (length 2,709). `TimeTok` maintains a similar model size of 9.42M–9.84M parameters. Although TimeVQVAE and Diffusion-TS are smaller on Nasdaq, `TimeTok` uses fewer parameters than all compared methods on HandOutlines while achieving strong generation performance.

E Coarse To Fine Reconstruction

We compare the coarse-to-fine *reconstruction* performance between individual tokenizers (I) trained on specific datasets and a foundation tokenizer (F) trained on the UTSD dataset, which contains time series from various domains.

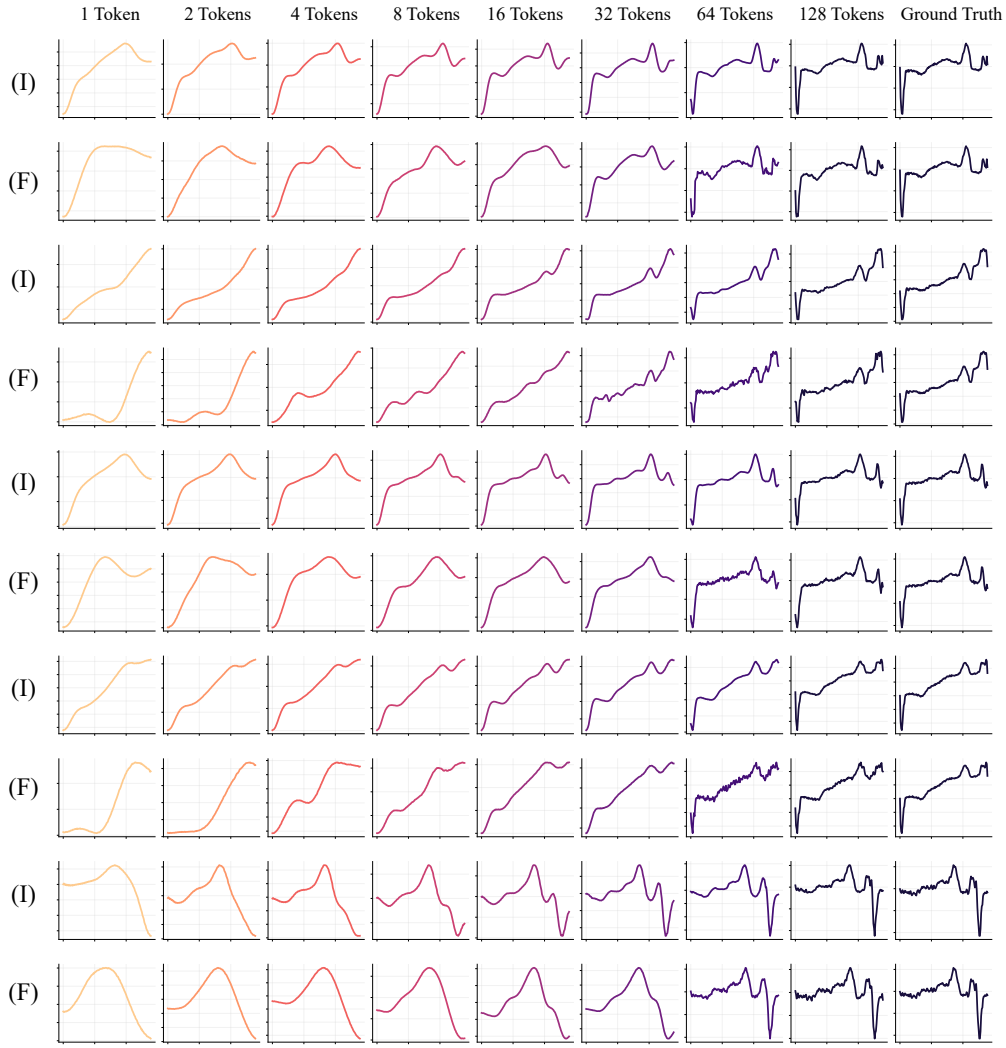


Figure 8: **ECG5000 Reconstruction.**

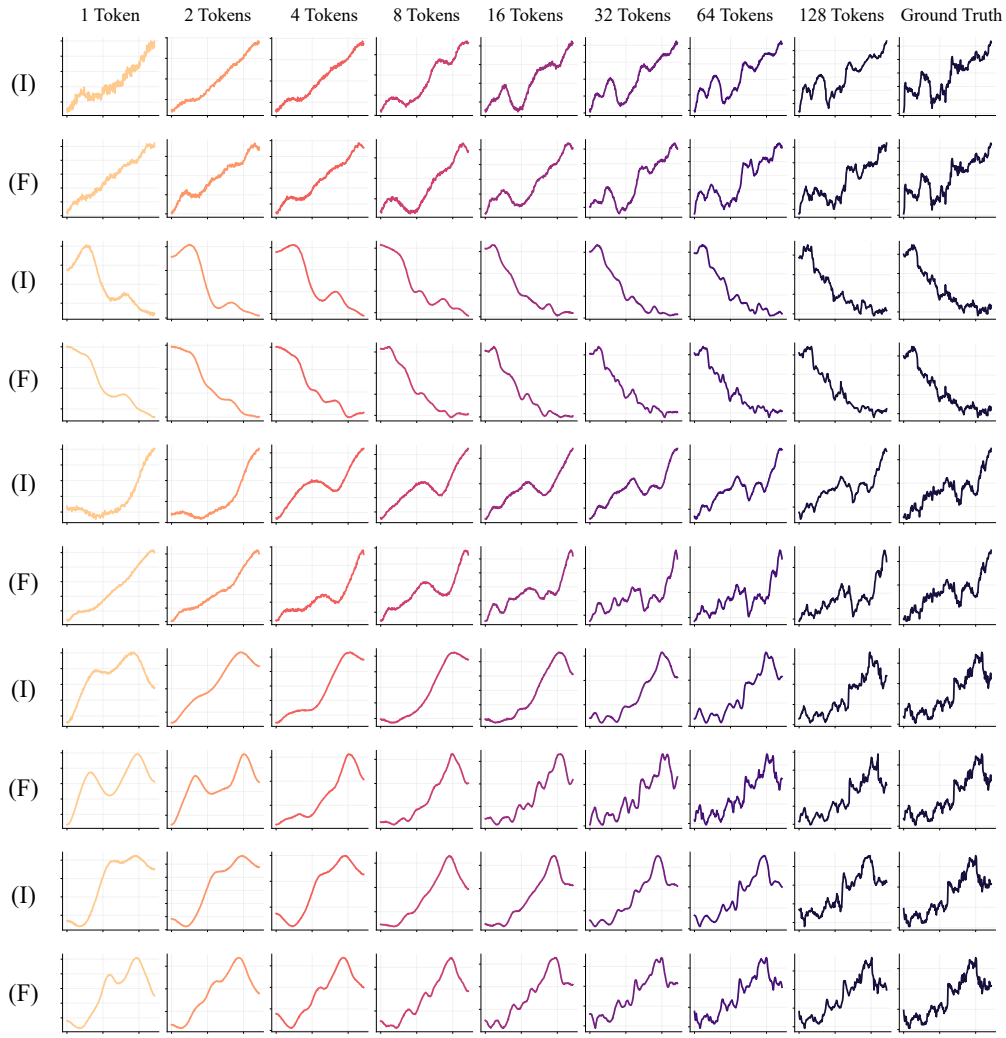


Figure 9: **Nasdaq Reconstruction.**

F GC-TSG Generation

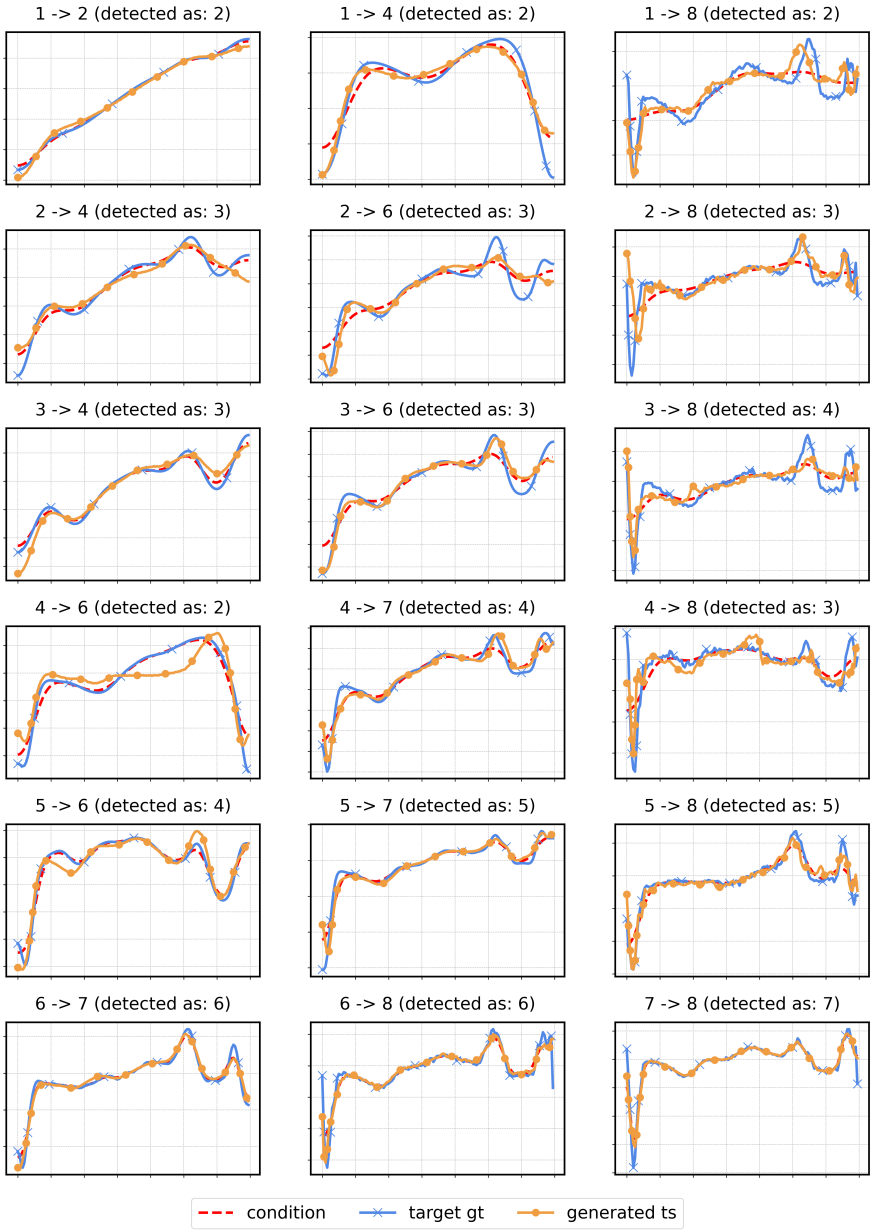


Figure 10: **GC-TSG of ECG5000.** We visualize the generation of $i \rightarrow j$ in ECG5000 dataset. Our level detection module first detects the granularity level of the sample. Using this sample as the condition, we generate time series at target granularity.

G Other Information Reducing Operations

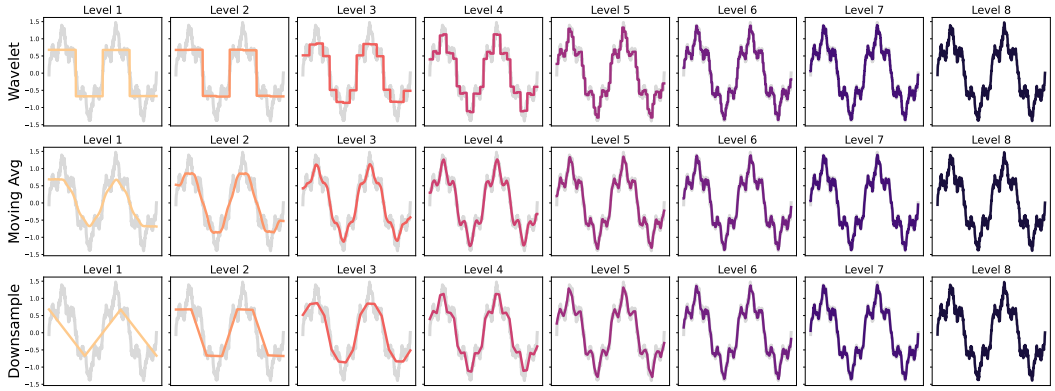


Figure 11: **Information Reducing Operators.** Visualization of alternative information reducing operators for each granularity level

Table 20: **Evaluation Results of Other Information Reducing Operators.** We report the CRPS_{sum} value at GC-TSG task.

	ECG5000, C2F $i \to j$			**Nasdaq, C2F $i \to j$**		
Operator	**TimeGAN**	**TimeVQVAE**	`TimeTok`	**TimeGAN**	**TimeVQVAE**	`TimeTok`
Wavelet	3.084±1.111	6.475±0.952	**2.354±0.916**	39.80±22.00	35.68±34.24	**17.29±13.05**
Moving Average	2.702±1.199	3.884±0.852	**2.080±0.905**	39.66±21.99	35.75±34.99	**17.13±12.91**
Downsample	2.686±1.126	3.496±0.998	**2.126±0.915**	39.51±22.15	34.63±34.14	**16.50±12.55**
Gaussian	1.737±0.737	3.693±0.858	**1.587±0.867**	39.11±22.14	35.32±34.30	**15.87±12.48**

In Table 20, we evaluate the GC-TSG performance of `TimeTok` on evaluation sets constructed with various information reducing operation functions ($\mathcal{R}_\ell$). Although `TimeTok` is trained exclusively with Gaussian kernel smoothing, it generalizes robustly to other operators such as wavelet transforms, moving averages, and down sampling. This robustness is particularly important for practical applications, as it demonstrates that `TimeTok` can effectively handle diverse coarse inputs conditions, such as user provided sketches and coarse outline.

H Use Case Analysis

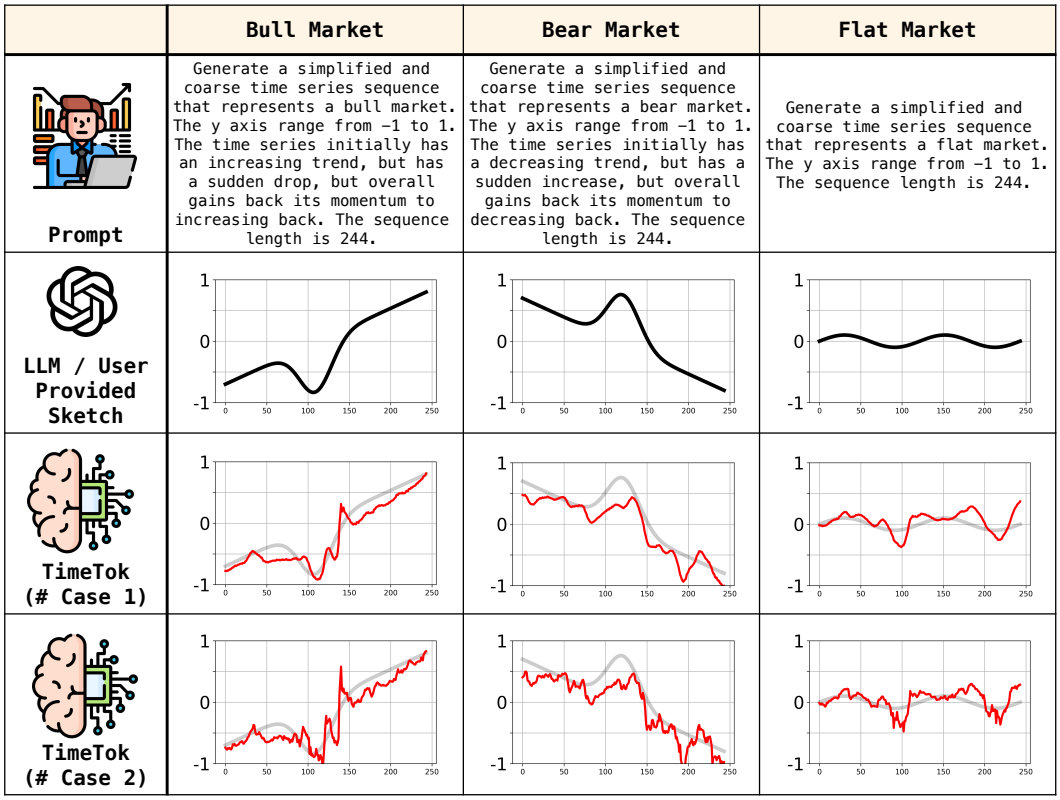


Figure 12: **Generating Fine-Grained Signals Conditioned on Coarse Outline: An Analysis in Financial Market Scenario.** For this analysis, we used `TimeTok` trained on the Nasdaq dataset. We employed ChatGPT to write code that generates the coarse signal, then used `TimeTok` to generate time series at various granularity levels, conditioned on the given coarse signal.

Complex time series are inherently difficult to describe using natural language due to their temporal structure and inherent randomness. In contrast, specifying a coarse outline of the time series is relatively straightforward: a user can simply draw the outline or ask an LLM to generate such signals or scenarios. This motivates an important application of `TimeTok`: generating fine-grained signals conditioned on a coarse outline. For instance, a financial analyst may want to model a bull market scenario, but with a slight drop in between. Such a coarse outline can be easily provided with an LLM as shown in the bull market scenario in Figure 12. Using this coarse outline, the analyst can use `TimeTok` to generate realistic financial bear market movement at various granularity levels, conditioned on the signal. We note that such generation is grounded on the actual distribution of the stock market (*i.e.,* `TimeTok`'s training data), differentiating with methods that simply add random noise to the coarse structure.

I The Use of LLMs in this work

In this paper, we employed Large Language Models (LLMs) for sentence restructuring to enhance clarity. We also used LLMs for the experiments in Appendix H.

J Token Usage Analysis on the Nasdaq Dataset

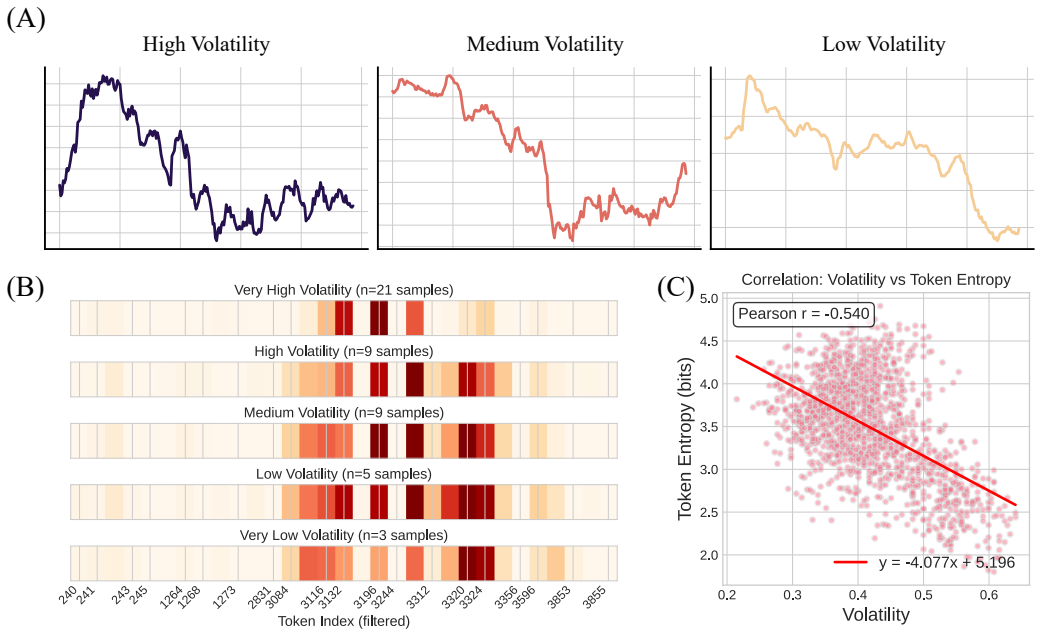


Figure 13: **Token Usage Pattern Analysis in Nasdaq. (A)** We quantify samples based on their volatility, and visualize three representative samples. **(B)** To narrow down the analysis space, we filter to samples that start with token index `3260`, which generally exhibit an increasing trend pattern. From these samples, we visualize a heatmap depicting the token usage pattern at the full sequence length (*i.e.,* 128 tokens). We observe that higher-volatility samples tend to utilize fewer tokens, whereas lower-volatility samples exhibit larger and more diverse token usage. **(C)** Understanding that such pattern occurs in the selected samples, we subsequently expand our focus to all samples. In the scatter plot, the x-axis represents the volatility, and the y-axis is the token usage entropy. We observe that such token-usage pattern occurs throughout the whole samples.

Our analysis focus on how the tokens in `TimeTok` are used to represent the given time series. To analyze token usage pattern in `TimeTok`, we used the Nasdaq synthetic dataset generated for the marginal experiment (the dataset used in Table 11). Specifically, we used samples generated at level 8 ($\ell = 8$), as they utilize the full tokens (*i.e.,* $M = 128$). We describe our analysis procedure in three stages.

(A). Given the time series, we first quantify the volatility of each data samples by measuring the Turning Point Rate [42], measuring the jaggedness of the data. In Figure 13-(A), we visualize samples based on the quantified measure. High volatility samples have high jaggedness, while low volatility samples have low jaggedness.

(B). We then tokenize the time series. To focus our token analysis, we identify token index `3260` as the most frequently occurring first token ($\ell = 1$) in the Nasdaq dataset and retain only the samples that begin with this token. Since the first token captures the coarsest pattern of the time series, we observe that token index `3260` generally corresponds to an increasing trend. From these samples, we visualize the used token indexes from all token sequences (*i.e.,* 128) using a heatmap. We observe that higher-volatility samples tend to use smaller subset of tokens, while lower-volatility samples have more diverse use of tokens.

(C). With this finding, we subsequently expand our analysis to the full sample space. We quantify the token usage pattern with token entropy, where a higher entropy value means more diverse use of tokens, and lower entropy corresponds to more selective use of tokens. From the scatter plot, we observe that a similar token usage pattern occurs throughout all samples.

In summary, we observe that samples with high volatility use a smaller subset of tokens, while lower volatility samples use a larger subset of tokens.

K Robustness Between Evaluators

We evaluate TSTR performance across linear (DLinear), recurrent (xLSTM), convolutional (InceptionTime), Transformer, and pretrained foundation-model evaluators on ECG5000 and ETTh1. For foundation-model evaluation, we use TimEE for classification without gradient updates and Chronos-2 for forecasting with LoRA fine-tuning. `TimeTok` achieves mean generator ranks of 1.4 on ECG5000 and 3.2 on ETTh1, supporting the robustness of the main findings across different evaluators.

Table 21: ECG5000 — F1 (higher is better).

	Linear (DLinear)	RNN (xLSTM)	CNN (InceptionTime)	Transformer	TS-FM (TIMEE)
RNN (TF)	0.326±0.003	0.356±0.010	0.382±0.009	0.340±0.014	0.378±0.000
TimeVQVAE	0.443±0.003	0.383±0.013	0.379±0.000	0.421±0.024	0.384±0.000
DiffusionTS	0.567±0.031	0.481±0.039	0.533±0.054	0.422±0.042	0.562±0.000
ImagenTime	0.626±0.054	0.504±0.037	0.488±0.093	0.554±0.034	0.608±0.000
TimeTok	0.555±0.020	0.512±0.045	0.583±0.016	0.585±0.033	0.622±0.000
Real	0.620±0.006	0.613±0.025	0.630±0.015	0.563±0.065	0.605±0.000

Table 22: ECG5000 — F1: method rank per evaluator (1 = best).

	Linear (DLinear)	RNN (xLSTM)	CNN (InceptionTime)	Transformer	TS-FM (TIMEE)	Mean rank
TimeTok	3	1	1	1	1	1.4
ImagenTime	1	2	3	2	2	2.0
DiffusionTS	2	3	2	3	3	2.6
TimeVQVAE	4	4	5	4	4	4.2
RNN (TF)	5	5	4	5	5	4.8

Table 23: ETTh1 — MSE (lower is better).

	Linear (DLinear)	RNN (xLSTM)	CNN (InceptionTime)	Transformer	TS-FM (Chronos-2)
RNN (TF)	0.104±0.000	0.112±0.001	0.111±0.001	0.107±0.001	0.104±0.000
TimeVQVAE	0.102±0.001	0.113±0.002	0.117±0.003	0.110±0.001	0.106±0.001
DiffusionTS	0.101±0.001	0.105±0.001	0.109±0.004	0.102±0.001	0.102±0.000
ImagenTime	0.099±0.000	0.073±0.002	0.013±0.003	0.090±0.005	0.104±0.000
TimeTok	0.103±0.000	0.107±0.001	0.102±0.002	0.105±0.002	0.105±0.001
Real	0.098±0.000	0.056±0.003	0.005±0.001	0.098±0.001	0.103±0.000

Table 24: ETTh1 — MSE: method rank per evaluator (1 = best).

	Linear (DLinear)	RNN (xLSTM)	CNN (InceptionTime)	Transformer	TS-FM (Chronos-2)	Mean rank
ImagenTime	1	1	1	1	2	1.2
DiffusionTS	2	2	3	2	1	2.0
TimeTok	4	3	2	3	4	3.2
RNN (TF)	5	4	4	4	3	4.0
TimeVQVAE	3	5	5	5	5	4.6

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Yes, we have faithfully reported our claims that reflect the paper's contribution.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations of this work are discussed in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

Justification: Justified with our answer.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a detailed elaboration on our architecture of `TimeTok`, and the experiments throughout the paper. We have also uploaded the codes.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We provide the dataset information in Appendix A.2, and will release the codes for `TimeTok`.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The details of our work are thoroughly discussed throughout the main paper and our appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We reported the standard deviation for three different random seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We stated our resource in Appendix D.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: Our research respects the NeruIPS Code of Ethics

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Broader impacts of this work are discussed in Appendix B.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper is an algorithmic paper.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We have appropriately cited the works.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: We do not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: Not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: Not relevant to our work.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: We provided our use of LLM in Appendix I.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.