Temporal-Scale Sensitivity in Time-Series Tokenization and Scale-Robust Token Estimation by Gated Sum

Berken Utku Demirel
Department of Computer Science
ETH Zürich, Switzerland
berken.demirel@inf.ethz.ch

Christian Holz
Department of Computer Science
ETH Zürich, Switzerland
christian.holz@inf.ethz.ch

Abstract

Transformers for temporal sequences require mapping sampled signals into sequences of vectors, called tokens. This is done by partitioning the sequence into local temporal windows (patches). We show that the tokenization step induces an approximation–estimation tradeoff. Smaller windows yield higher-variance statistical estimates because each token is supported by fewer samples, while larger windows stabilize estimation but increase the amount of information that must be summarized. Motivated by this tradeoff, we propose a lightweight tokenizer that constructs multiple estimates for each token and sums them, without changing the Transformer backbone. Across 12 datasets spanning four modalities, our method improves performance and reduces sensitivity to temporal scale across tasks.

1 Introduction

Transformer architectures operate on sequences of vectors [1]. In natural language processing, these vectors are constructed from an inventory of discrete units, such as words or subwords [2, 3], commonly referred to as tokens [4–6]. Because language is a symbolic communicative system [7, 8], its input already carries a learned discrete organization [9, 10] and substantial structural redundancy [11, 12], which provide a natural substrate for tokenization and sequence modeling [13–16]. Although token granularity in language is itself a choice [2, 4], models still operate over a symbolic sequence shaped by historical evolution and by pressure for efficient communication [17, 18].

Time series, however, lack such intrinsic symbolic units. Temporal signals are physical processes recorded through discrete sampling [19, 20]. Because individual samples are scalar values—corrupted by noise—they carry little semantic meaning on their own [21, 22]. To apply attention, the model must first define its own tokens by partitioning the discretely sampled signal into local windows (patches), mapping each window to a vector [23, 24]. Unlike in language, where tokenization acts on an existing symbolic sequence [4, 25], tokenization for time series defines the representational units themselves, thereby setting the temporal scale of the model and influencing how noise is handled.

We show that tokenization of time-series induces an approximation-estimation tradeoff shaped by sampling rate and window-level complexity. Smaller windows preserve the fine-scale structure of the signal but contain fewer sampled points, leading to higher variance of statistical features. Conversely, larger windows improve estimation, but they also increase the amount of within-window variation that must be represented by a single token, making the approximation problem harder. We introduce a lightweight tokenization method that improves estimation within windows to reduce sensitivity to window choice while leaving the architecture unchanged. Our contributions can be summarized as:

1. Providing a theoretical explanation of the tradeoff inherent in temporal sequence tokenization.
2. Empirically validating the derived theoretical tradeoff across real-world and synthetic datasets.
3. Proposing a lightweight tokenization method without changing the Transformer model.

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

2 Motivating Example

As an example of windowing effects in Transformers, we consider heart rate (HR) estimation from photoplethysmogram (PPG), a task related to periodic structure and frequency estimation in noisy non-stationary signals [26]. We use a Transformer [27], with six layers and four heads, on datasets with similar HR distributions and identical sampling rate (25 Hz) and duration (8-second) [28]. For each dataset, we train and evaluate the model separately, varying only the tokenization window length.

Figure 1 reveals an unexpected pattern. The best window size is not stable across datasets. Although the target frequency band (heart rate) and periodic pattern are the same, window sizes that perform well on one dataset can lead to degradation on another, by as much as 95% (e.g., from 1.76 to 3.43). In the next section, we explain this phenomenon using tools from estimation theory and signal processing.

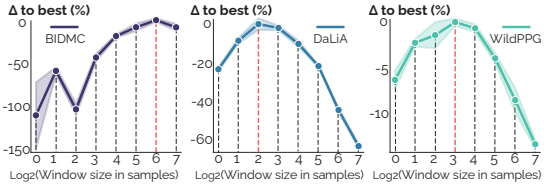

Figure 1: Mean absolute error (MAE) as a function of window size. The best window is normalized to 0, and negative values indicate the relative increase in error compared with the best achieved performance.

3 Tokenization of Temporal Sequences

3.1 Notation and Preliminaries

We use lowercase letters (e.g., x) for scalars, bold lowercase letters (e.g., $\mathbf{x}$) for vectors such as time series, and bold uppercase letters (e.g., $\mathbf{X}$) for matrices. We consider temporal signals with sampling rate f_s Hz. Let T and Δ denote the sequence duration and window length, in seconds, respectively. We use *window* and *patch* interchangeably for a temporal segment of duration Δ, while *token* refers to its learned vector representation. Each window contains $M_\Delta = f_s\Delta \in \mathbb{N}$ samples, and a sequence of duration T is partitioned into $N = T/\Delta$ windows. We write $\|\mathbf{x}\|$ for a vector norm.

3.2 Windowing

Let $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times T f_s}$ denote a sampled input sequence of duration T with C_{in} channels. We partition $\mathbf{x}$ into N windows $I_1, \ldots, I_N$, each of duration Δ. The resulting windows are denoted by $\mathbf{w}_1, \ldots, \mathbf{w}_N$, where $\mathbf{w}_i := \mathbf{x}|_{I_i} \in \mathcal{W}_\Delta$. A parameterized map $\hat{\phi}_\Delta : \mathcal{W}_\Delta \to \mathbb{R}^d$ maps each window $\mathbf{w}_i$ to a d-dimensional token $\mathbf{z}_i := \hat{\phi}_\Delta(\mathbf{w}_i)$, yielding the sequence $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{N \times d}$. This sequence is then combined with positional encodings and used as input to the Transformer network.

3.3 Window Space and Error Decomposition

To analyze how well the learned tokens approximate an idealized target representation, we define an oracle map $\phi^\star_\Delta : \mathcal{W}_\Delta \to \mathbb{R}^d$, where $\phi^\star_\Delta(\mathbf{w})$ denotes an abstract target feature of the window for the downstream task. The corresponding target and empirical statistics for a window are $\theta^\star_i := \phi^\star_\Delta(\mathbf{w}_i)$ and $\hat{\theta}_i := \hat{\phi}_\Delta(\mathbf{w}_i)$, respectively. Uniform pooling combines all windows equally, while attention can place more weight on a few windows. We model these weighting patterns through $\theta^\star = \sum_{i=1}^N \omega_i \theta^\star_i$ and $\hat{\theta} = \sum_{i=1}^N \omega_i \hat{\theta}_i$, where $\omega_i \geq 0$ and $\sum_i \omega_i = 1$. We hold the weights and window targets fixed under the estimation and optimization randomness. This isolates how token errors combine for a given weighting pattern. Following the bias–variance convention [29], we separate each window error into its mean b_i and a centered fluctuation ξ_i, as in Equation 1.

$$e_i = \hat{\theta}_i - \theta^\star_i = \hat{\phi}_\Delta(\mathbf{w}_i) - \phi^\star_\Delta(\mathbf{w}_i), \quad \text{where} \quad b_i := \mathbb{E}[e_i], \quad \xi_i := e_i - b_i, \quad \text{and} \quad \mathbb{E}[\xi_i] = 0 \tag{1}$$

Figure 2 summarizes this abstraction. Two example windows ($\mathbf{w}_1$ and $\mathbf{w}_2$) are mapped into the d-dimensional token space, together with their oracle targets and the associated error decomposition.

To study the effect of the window size Δ, we analyze the mean-squared estimation error [30, 31]. We treat $\hat{\phi}_\Delta(\mathbf{w}_i)$ as a noisy estimator of the target

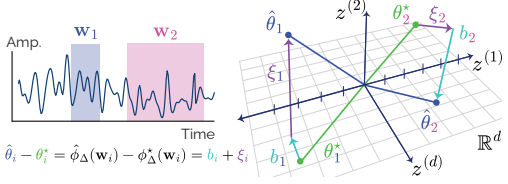

Figure 2: A sequence and its token representations with error decomposition $\hat{\theta}_i - \theta^\star_i = b_i + \xi_i$.

window statistic and take expectation with respect to this estimation randomness, conditional on the windows. Each window error contributes to the segment according to its weight ω_i. We write the resulting bias and residual as $\bar{b} = \sum_i \omega_i b_i$ and $\bar{\xi} = \sum_i \omega_i \xi_i$. The segment-level error therefore separates into the two terms in Equation 2.

$$\hat{\theta} - \theta^\star = \bar{b} + \bar{\xi}, \qquad \mathbb{E}\left[\|\theta^\star - \hat{\theta}\|^2\right] = \|\bar{b}\|^2 + \mathbb{E}\left[\|\bar{\xi}\|^2\right], \tag{2}$$

A fixed weighted sum of centered residuals remains centered, so $\mathbb{E}[\bar{\xi}] = \sum_i \omega_i \mathbb{E}[\xi_i] = 0$. Because $\bar{b}$ is deterministic, this also implies $\mathbb{E}\langle \bar{b}, \bar{\xi} \rangle = 0$, which removes the mixed term from the error.

Bound the bias. To make the bias term explicit, we introduce an operator $T_\Delta : \mathcal{W}_\Delta \to \mathcal{S}_\Delta \subset \mathcal{W}_\Delta$, where $\mathcal{S}_\Delta$ denotes a class of surrogate windows that are representable by a single token at scale Δ. Here, T_Δ is an abstract analysis operator used only for the error decomposition. It represents the within-window information that can be retained by one token at scale Δ (see Fig. 3). For example, if a token retains only the window mean, then $T_\Delta \mathbf{w}$ is the constant function on that window equal to its average. If the token retains coefficients in a finite-dimensional family, then $T_\Delta \mathbf{w}$ is the corresponding reconstruction in that family, such as low-order polynomials or trigonometric expansions [32].

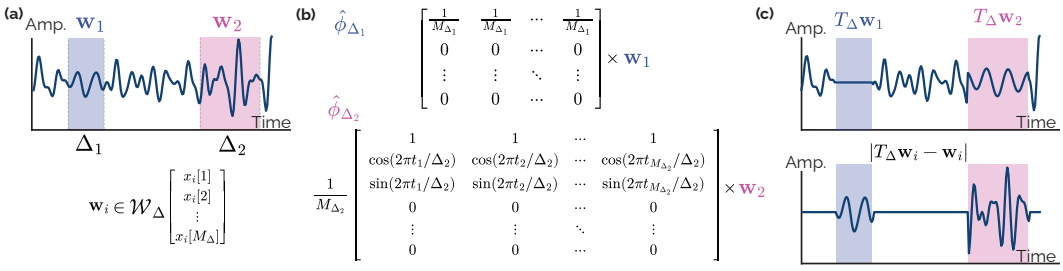

Figure 3: **Illustration of the operator T_Δ and the induced approximation error. (a)** A sequence with two windows $\mathbf{w}_1$ and $\mathbf{w}_2$ of durations Δ_1 and Δ_2, respectively. Each window is a vector $\mathbf{w}_i \in \mathcal{W}_\Delta$, with $M_\Delta = f_s \Delta$ samples. **(b)** Two example window-to-token maps. The map $\hat{\phi}_{\Delta_1}$ is averaging, which retains only the window mean. $\hat{\phi}_{\Delta_2}$ is represented by a richer map whose rows correspond to sinusoidal functions. This map retains more window variation. **(c)** The T_Δ is visualized in the time domain. For each window, $T_\Delta \mathbf{w}_i$ denotes the surrogate signal retained by one token. The first window is approximated by a constant, whereas the second retains oscillatory structure, reflecting a richer approximation class. The lower plot shows the error $|T_\Delta \mathbf{w}_i - \mathbf{w}_i|$. The error is small when the chosen surrogate class matches the structure of the signal and large when the window contains variations that the token cannot represent. The figure therefore illustrates why we introduce T_Δ. It provides a principled abstraction of the information bottleneck induced by tokenization and clarifies how the approximation burden depends on window size and the richness of the token map.

To connect the bias to the information retained by a token, we introduce the intermediate target $\tilde{\theta}_i := \phi^\star_\Delta(T_\Delta \mathbf{w}_i)$. This evaluates the target feature on the retained approximation of the window. We assume $\mathbb{E}[\hat{\theta}_i] = \tilde{\theta}_i$ to isolate the systematic error caused by this approximation. Under this assumption, the local error separates into the two components in Equation 3.

$$\hat{\theta}_i - \theta^\star_i = \underbrace{\left(\hat{\phi}_\Delta(\mathbf{w}_i) - \phi^\star_\Delta(T_\Delta \mathbf{w}_i)\right)}_{\xi_i} + \underbrace{\left(\phi^\star_\Delta(T_\Delta \mathbf{w}_i) - \phi^\star_\Delta(\mathbf{w}_i)\right)}_{b_i}, \tag{3}$$

The bias $b_i = \tilde{\theta}_i - \theta^\star_i$ measures how the target feature changes when the original window is replaced by its retained approximation. The residual $\xi_i = \hat{\theta}_i - \tilde{\theta}_i$ describes fluctuations around this intermediate target. Appendix A treats any additional mean error of the learned estimator. Representative sources of b_i and ξ_i are listed in Appendix A.3.

Proposition 3.1 (Token-induced bias component)**.** *Given an oracle map $\phi^\star_\Delta : \mathcal{W}_\Delta \to \mathbb{R}^d$ that is Lipschitz with respect to the window norm $\|\cdot\|_{\mathcal{W}_\Delta}$, i.e., $\|\phi^\star_\Delta(\mathbf{u}) - \phi^\star_\Delta(\mathbf{v})\| \le L_\phi \|\mathbf{u} - \mathbf{v}\|_{\mathcal{W}_\Delta}$, and a token-induced approximation operator T_Δ such that the retained window approximation satisfies $\|\mathbf{w} - T_\Delta \mathbf{w}\|_{\mathcal{W}_\Delta} \le C_{\mathrm{app}} \Delta^r$, the squared norm of the bias term $\|\bar{b}\|^2$ is bounded by*

$$\|\bar{b}\|^2 \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r},$$

where C_{app} measures how well the token-retained representation approximates the original window in $\|\cdot\|_{\mathcal{W}_\Delta}$, and r specifies how that approximation improves as the window size Δ decreases.

Proof. By definition, $b_i = \phi^\star_\Delta(T_\Delta \mathbf{w}_i) - \phi^\star_\Delta(\mathbf{w}_i)$. Given $\|b_i\| \le L_\phi \|T_\Delta \mathbf{w}_i - \mathbf{w}_i\|_{\mathcal{W}_\Delta} \le L_\phi C_{\mathrm{app}} \Delta^r$. Taking the weighted sum and applying the triangle inequality gives

$$\|\bar{b}\| = \left\| \sum_{i=1}^N \omega_i b_i \right\| \le \sum_{i=1}^N \omega_i \|b_i\| \le L_\phi C_{\mathrm{app}} \Delta^r.$$

Squaring both sides yields $\|\bar{b}\|^2 \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r}$. □

Examples of admissible classes that give specific values of r (e.g., Hölder-smooth or Sobolev signals) are provided in Appendix A, where we also derive approximation bounds for common tasks.

Proposition 3.1 shows why a good window approximation also preserves the target feature. The Lipschitz condition ensures that a small change in the window produces a small change in that feature. The constants C_{app} and r describe how the approximation bound changes with window duration.

Proposition 3.1 is also aligned with the classical representation-learning principle that an obtained representation should have the task-relevant information [33]. Here, that principle appears concretely through the approximation error $\|\mathbf{w} - T_\Delta \mathbf{w}\|$, which directly controls the bias term.

For $r > 0$, shorter windows reduce the approximation bound because each token represents a shorter part of the signal. However, each token also has fewer samples from which to estimate its target feature. We therefore need to consider how this loss of sample support affects estimation variance.

Bound the variance. A shorter window contains fewer samples, which can make its target statistic harder to estimate. We express this through the local estimation law $\mathbb{E}\|\xi_i\|^2 \le c M_\Delta^{-p}$, where $M_\Delta = f_s \Delta$ and $c > 0$ and $p > 0$ are fixed over the window sizes considered. The exponent p describes how quickly additional samples improve the estimate and depends on the statistic, estimator, and noise model. For a sample mean with independent finite-variance noise, the variance decreases inversely with the number of samples, giving $p = 1$. Structured parameters such as single-tone frequency can admit faster rates, including $p = 3$ [34].

Proposition 3.2 (Variance with cross-window dependence)**.** *Let $\bar{\xi} = \sum_{i=1}^N \omega_i \xi_i$, with fixed weights $\omega_i \ge 0$ and $\sum_i \omega_i = 1$. Suppose the centered residuals satisfy $\mathbb{E}\|\xi_i\|^2 \le c M_\Delta^{-p}$. Write $a := c f_s^{-p}$ and $N_{\mathrm{eff}}(\Delta) := (\sum_i \omega_i^2)^{-1}$. The segment-level residual variance satisfies Equation 4.*

$$\mathbb{E}\|\bar{\xi}\|^2 \le \frac{a\Delta^{-p}}{N_{\mathrm{eff}}(\Delta)} + 2 \sum_{i<j} \omega_i \omega_j \mathbb{E}\langle \xi_i, \xi_j \rangle. \tag{4}$$

Proof. Expanding the squared norm separates the variance and covariance terms.

$$\mathbb{E}\|\bar{\xi}\|^2 = \sum_i \omega_i^2 \mathbb{E}\|\xi_i\|^2 + 2 \sum_{i<j} \omega_i \omega_j \mathbb{E}\langle \xi_i, \xi_j \rangle.$$

The local estimation law bounds each residual variance by $a\Delta^{-p}$, so the first term is at most $a\Delta^{-p} \sum_i \omega_i^2 = a\Delta^{-p}/N_{\mathrm{eff}}(\Delta)$. □

The quantity N_{eff} describes how widely the aggregation weights are spread. Equal weights on K contributing windows give $N_{\mathrm{eff}} = K$, while placing all weight on one window gives $N_{\mathrm{eff}} = 1$. The covariance term captures whether errors in different windows reinforce or cancel one another. Positive covariance can weaken the benefit of combining estimates because several windows repeat part of the error. Overlapping windows can exhibit this dependence because they share noisy samples. Appendix A gives the overlapping-window derivation and examples of local estimation rates.

Corollary 3.3 (Uncorrelated residuals)**.** *Under the assumptions of Propositions 3.1 and 3.2, suppose $\mathbb{E}\langle \xi_i, \xi_j \rangle = 0$ for $i \ne j$. The cross-window terms vanish, leaving only the weighted sum of individual residual variances. Combining their bounds with the bias contribution gives the total error bound.*

$$\mathbb{E}\|\hat{\theta} - \theta^\star\|^2 \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r} + \frac{a\Delta^{-p}}{N_{\mathrm{eff}}(\Delta)}. \tag{5}$$

Uniform weights give $N_{\text{eff}} = N = T/\Delta$. Equation 6 states the resulting uniform-pooling bound.

$$\mathbb{E}\|\hat{\theta} - \theta^\star\|^2 \le L_\phi^2 C_{\text{app}}^2 \Delta^{2r} + \frac{a}{T}\Delta^{-(p-1)}. \tag{6}$$

For uniform pooling, shortening the windows makes each estimate noisier but also provides more estimates to average. At $p = 1$, these two effects cancel in the estimation bound. If only a few windows receive substantial weight, the additional windows offer less averaging benefit. The estimation term can therefore grow as windows become shorter even when $p = 1$.

To describe intermediate weighting patterns, consider $N_{\text{eff}}(\Delta) = C(T/\Delta)^\nu$, with constants $C > 0$ and $0 \le \nu \le 1$. Substituting this relation into the estimation term gives Equation 7.

$$\frac{a\Delta^{-p}}{N_{\text{eff}}(\Delta)} = \frac{a}{CT^\nu}\Delta^{-(p-\nu)}. \tag{7}$$

When $p > \nu$, shorter windows increase the estimation term while reducing the approximation term. For positive coefficients and $r > 0$, the resulting two-term bound has an interior minimum over $\Delta > 0$. Uniform averaging corresponds to $\nu = 1$, recovering the condition $p > 1$. A constant effective number of windows corresponds to $\nu = 0$, allowing such a minimum for any $p > 0$, including $p = 1$. With correlated residuals, the covariance term also affects this balance. An observed optimum in classification performance does not identify the local estimation exponent p. Overall, Δ sets the temporal scale at which information is summarized before being passed to the Transformer.

3.4 Revisiting the Motivating Example

The theory argues that the optimal window size is governed by the tradeoff between approximation and estimation. To relate this to the initial example, we compute dataset-level statistics that act as empirical proxies for these two effects. Therefore, we calculate the signal-to-noise ratio (SNR), using heart rate labels, as a proxy for estimation quality, and spectral entropy (SE) together with the Jensen–Shannon divergence (JSD) between neighboring segments as approximations for local variability, i.e., how difficult the signal is to compress within a single window [35].

The dataset-level statistics are consistent with the theory. BIDMC [36], which has the highest SNR and the lowest variation (see Fig. 4 where the y-axis is the probability density of the features), performs better with larger windows, whereas DaLiA and WildPPG [28], which exhibit much stronger variation, perform better with smaller windows.

Overall, the example supports the theory, and Appendix I further discusses how the experiments relate to the theory. Signals with slow variation favor longer windows because the gain in estimation outweighs the approximation cost, whereas signals with stronger variations favor shorter windows because the approximation burden of summarizing a long non-stationary window becomes dominant.

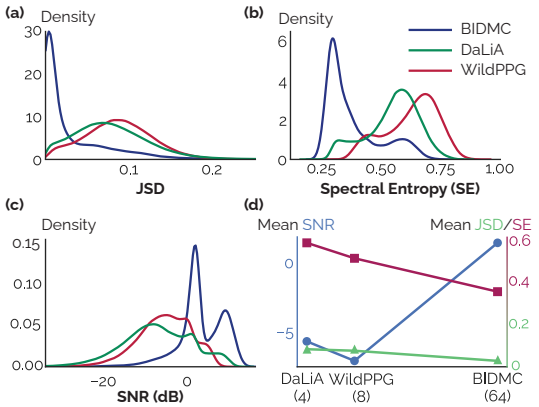

Figure 4: **(a)–(c)** The calculated features across datasets. **(d)** Relationship between the optimal window size (x-axis, M_Δ) and feature means (y-axes).

4 Learning-based Multitaper

We show that tokenization using a single window Δ induces a tradeoff. Therefore, we introduce a *multitaper*-inspired tokenizer that constructs several estimates of each token and sums them through gates. For each view m, let $\bar{\mathbf{x}}_n^{(m)} \in \mathbb{R}^{C_{\text{in}}k_m}$ denote the receptive-field vector centered at location n, obtained by stacking the k_m multichannel samples used by that view. Let $\bar{\mathbf{W}}^{(m)} \in \mathbb{R}^{d\times C_{\text{in}}k_m}$ denote the corresponding matricized kernel. The token estimate produced by view m is

$$\mathbf{z}_n^{(m)} = \bar{\mathbf{W}}^{(m)}\bar{\mathbf{x}}_n^{(m)} \in \mathbb{R}^d \tag{8}$$

We sum the estimates through learned gates. Let $g_m \in \mathbb{R}$ be the gate parameter for view m, and define $\alpha_m = \sigma(g_m)$, where $\sigma(\cdot)$ is the sigmoid function. The obtained token is $\mathbf{z} = \sum_{m=1}^{K} \alpha_m \mathbf{z}^{(m)}$. A step-by-step algorithmic summary and more details are given in Appendix D.1.

In classical multitaper estimation [37], fixed tapered views of the same finite sequence are averaged to reduce sensitivity to a single windowing choice. Our method follows this principle, but replaces fixed tapers with learned temporal analysis views. Unlike ordinary multi-scale convolution, these views do not preserve separate scale-specific branches; they produce alternative estimates of the same token, which are summed before entering the Transformer.

Thus, the proposed method does not remove the approximation-estimation tradeoff; it adapts to it. Rather than fixing one window size, it forms multiple views and learns their summation to obtain a richer first-stage representation on the same grid without changing the original architecture.

From a representation perspective, the method replaces a single analysis map $\hat{\phi}_\Delta : \mathcal{W}_\Delta \to \mathbb{R}^d$ with a family of analysis maps $\hat{\phi}_\Delta^{(m)} : \mathcal{W}_\Delta \to \mathbb{R}^d$, $m = 1, \ldots, K$, and defines the multitaper token by

$$\mathbf{z}_n = \hat{\phi}_\Delta^{\mathrm{MT}}(\mathbf{w}_n) = \sum_{m=1}^{K} \alpha_m \hat{\phi}_\Delta^{(m)}(\mathbf{w}_n) \tag{9}$$

A standard normalization is then applied before passing the token sequence to the transformer. The role of the multitaper is to reduce sensitivity to any single tokenization geometry by summing several finite-sample estimates. In this view, tokenization becomes a learned finite-sample estimation problem whose design is guided by the approximation-estimation tradeoff, rather than a fixed preprocessing.

5 Experimental Setup

5.1 Datasets

We conducted experiments on 12 datasets across four tasks, including heart rate (HR) estimation from PPG, activity recognition using inertial measurements (IMUs), cardiovascular disease classification from electrocardiogram (ECG) and sleep stage classification from electroencephalography (EEG). We provide brief descriptions of each dataset below. Additional details are available in Appendix B.

Heart rate. We used the IEEE Signal Processing Cup in 2015 (IEEE SPC) [38], DaLiA [39], WildPPG [28], and BIDMC [36]. We used leave-one-session-out (LOSO) cross-validation, which evaluates models on subjects/sessions not used for training, except for BIDMC, where we used 10-fold cross-validation.

Activity recognition. We used UCIHAR [40], HHAR [41], and USC [42] for activity recognition from phones or wearable devices, depending on the dataset, and Clemson [43] for step counting. We evaluated cross-person generalization by testing each model on previously unseen subjects.

Cardiovascular disease (CVD) classification. We conducted experiments on the PhysioNet/Computing in Cardiology Challenge 2017 (PhysioNet2017) [44], PTB-XL [45] and Chapman [46] datasets. For PTB-XL, we follow the recommended split based on the official stratified folds. For all datasets, we enforce subject-level splitting to prevent subject overlap across training, validation, and test sets.

Sleep stage classification. We used the Sleep-EDFx [47], from PhysioBank [48], which includes 197 PSG recordings, where we used a Fpz-Cz EEG channel with a sampling rate of 100 Hz. We used non-overlapping 30 s windows and used a 6/2/2 split [49] for training, validation, and testing.

5.2 Baselines

We group the baselines by whether they isolate tokenization or couple it with architectural changes.

Tokenization-only baselines. These include tokens formed from linear embeddings with non-overlapping (PatchTST) and overlapping (ov.) windows as standard patch-based baselines [23]. We also include convolutional kernels as tokenizers [50], and TOTEM [51], which is a baseline that learns whole-signal discrete tokens using a tailored VQ-VAE [52] with a reconstruction objective.

Joint tokenization–architecture baselines. We compare with PathFormer [53], MedFormer [54], ScaleFormer [55], MTST [56], Kairos [57], and Peri-midFormer [58]. These methods couple

tokenization with additional architectural modifications. We include them to provide a broader comparison. Further descriptions and details are provided in Appendix C and D.3.

5.3 Implementation

We used a pre-norm Transformer encoder [1] with six layers, four attention heads, embedding dimension 64, MLP hidden dimension 128, and dropout 0.1. For the multitaper tokenizer, we used distinct multiple (2 or 3) kernels k_m selected from $\{5, 9, 17, 33, 65, 129\}$, depending on the task. All regular Transformer models were trained with a batch size of 512 using AdamW [59] with a learning rate of 10^{-3}. Early stopping was applied based on the validation loss with a patience of 25 epochs. Reported results are the mean and standard deviation over three runs with different random seeds using the best model in the validation set. Further details and settings are provided in Appendix D, and Appendix E shows that the method is robust to the hyperparameters.

6 Results

Tables 1 to 3 present the results. Overall, our method achieves the strongest performance in 10 of the 12 datasets while remaining competitive in the other two. These gains are obtained with a lightweight tokenizer that preserves the original architecture. Importantly, hyperparameters are selected at the task level from a small subset rather than tuned separately for each dataset. Empty entries indicate settings in which the corresponding model collapsed to the majority or mean class prediction consistently across seeds, and are therefore omitted from quantitative comparison.

Since we evaluate methods on 12 datasets, three metrics, and multiple random seeds, we further assess statistical significance using per-task ranks, followed by a Friedman test with Nemenyi post-hoc comparisons. Our method achieves the best average rank and significantly outperforms the baselines. The critical-difference diagram and results are provided in Appendix E.2.

Table 1: Performance comparison of tokenization methods for heart rate estimation using PPG.

Method	IEEE SPC			WildPPG			DaLiA			BIDMC		
	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑
Linear	10.11±0.31	17.77±0.54	68.89±2.25	9.92±0.04	17.82±0.03	41.50±0.27	5.67±0.11	12.03±0.18	78.06±0.52	1.76±0.02	4.51±0.07	89.67±0.06
Linear (ov.)	8.10±0.11	16.74±0.35	73.57±1.07	9.96±0.04	17.91±0.44	41.19±0.37	5.90±0.10	11.94±0.29	78.30±0.87	1.67±0.05	4.11±0.40	89.62±2.23
PathFormer	18.52±1.08	24.66±1.50	26.70±9.72	13.95±0.37	19.18±0.33	14.16±2.66	—	—	—	2.70±0.15	5.30±0.20	89.48±0.65
MedFormer	7.80±0.34	15.60±0.59	72.86±2.16	10.08±0.01	17.84±0.02	39.98±0.38	5.83±0.10	12.62±0.19	75.88±0.41	2.21±0.16	5.22±0.53	85.24±3.50
Totem	18.79±0.44	24.54±0.71	37.07±8.28	—	—	—	—	—	—	4.58±0.13	8.04±0.12	72.43±1.31
PerimidFormer	23.06±1.03	30.96±1.31	14.27±2.04	14.61±0.38	22.66±0.54	19.82±2.44	17.67±0.23	24.94±0.46	28.63±1.36	9.58±0.48	9.95±0.50	20.23±4.52
ScaleFormer	12.88±0.06	17.15±0.19	69.79±0.73	10.73±0.16	18.13±0.16	35.29±1.76	6.65±0.71	13.09±0.21	74.26±0.65	3.04±0.07	5.26±0.07	89.51±0.18
MTST	17.35±0.34	22.68±0.59	60.28±1.26	13.29±0.10	18.43±0.32	18.96±2.27	9.90±0.64	16.45±1.14	68.12±1.35	3.95±0.29	9.14±0.73	65.43±0.06
Kairos	8.26±0.32	16.30±0.53	73.89±1.25	10.21±0.09	18.00±0.13	39.39±0.67	6.08±0.05	12.58±0.08	75.29±0.03	1.96±0.20	4.87±0.58	88.03±3.88
Multitaper	**6.50**±0.32	**12.80**±0.54	**82.32**±1.13	**9.75**±0.06	**17.63**±0.05	**42.15**±0.19	**5.42**±0.10	**11.50**±0.10	**78.58**±0.13	**1.58**±0.08	**4.04**±0.22	**91.10**±1.03
– w/o weights	7.03±0.15	13.57±0.26	80.92±1.30	9.92±0.02	17.81±0.02	41.70±0.09	5.63±0.12	11.90±0.06	77.50±0.30	1.59±0.06	4.08±0.19	91.03±0.44
– single-kernel	13.17±0.44	19.23±0.66	67.59±1.51	10.03±0.09	17.85±0.15	41.33±0.74	5.77±0.15	11.93±0.29	77.53±0.57	2.38±0.18	5.61±0.50	85.44±3.11
– same k_m	10.50±0.21	16.60±0.34	72.44±0.34	9.93±0.03	17.85±0.02	41.62±0.26	5.89±0.12	11.96±0.28	76.68±0.78	2.60±0.04	5.14±0.14	88.71±0.60

Table 2: Performance comparison of tokenization methods for activity and step using IMU.

Method	UCIHAR			HHAR			USC			Clemson		
	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	MAE ↓	RMSE ↓	ρ ↑
Linear	87.62±0.68	87.39±0.50	86.35±0.78	81.06±0.67	80.13±0.57	80.19±0.53	67.40±5.27	63.66±6.12	65.93±6.02	3.22±0.15	4.41±0.18	11.74±0.11
Linear (ov.)	87.95±0.75	87.87±0.85	87.01±1.02	**82.62**±0.95	**81.77**±0.84	**82.36**±1.07	71.83±0.80	69.13±1.23	70.47±1.40	1.96±0.13	3.24±0.18	59.28±4.56
PathFormer	70.12±0.35	64.89±3.66	66.16±4.51	50.85±4.72	46.23±5.67	47.32±5.79	51.47±2.26	48.14±2.21	48.77±2.55	2.39±0.26	3.59±0.28	44.92±6.55
MedFormer	83.98±5.50	83.24±5.57	82.58±6.98	77.26±1.15	76.11±1.42	75.96±1.58	67.89±3.09	68.45±3.22	66.19±3.60	1.91±0.21	3.26±0.16	57.57±6.09
Totem	79.87±0.29	80.12±0.07	80.07±0.16	—	—	—	62.54±3.06	68.42±0.40	67.03±1.26	—	—	—
PerimidFormer	80.00±8.13	78.96±9.12	79.13±9.04	68.07±3.53	65.93±3.79	66.82±3.64	59.90±6.67	56.60±6.38	57.88±7.00	3.41±0.13	4.46±0.18	5.67±0.31
ScaleFormer	88.54±2.63	87.95±3.14	87.52±3.33	81.66±0.98	80.44±0.88	81.22±0.93	71.90±2.30	65.67±1.32	68.24±1.96	3.46±0.20	4.78±0.27	7.16±5.40
MTST	73.14±1.47	71.07±1.62	70.22±1.59	72.67±1.04	70.89±0.83	71.36±1.14	59.56±2.34	54.28±2.60	56.26±3.26	3.80±0.67	4.52±0.73	10.36±1.97
Kairos	69.38±0.97	67.15±0.49	68.28±0.04	70.20±1.15	68.58±1.03	69.67±0.86	54.86±2.02	51.09±3.02	52.16±2.46	**1.61**±0.08	3.03±0.15	60.40±4.97
Multitaper	**91.37**±2.49	**91.30**±2.23	**90.89**±2.68	80.12±1.50	79.23±0.69	80.87±0.68	**74.69**±1.78	**71.60**±1.99	**73.25**±2.12	1.68±0.46	**2.92**±0.35	**61.53**±1.90
– w/o weights	92.52±3.35	92.14±3.63	92.16±3.71	78.52±1.64	77.53±1.48	77.90±1.80	74.51±2.04	70.75±2.46	72.86±2.52	1.70±0.45	2.96±0.33	60.26±8.22
– single-kernel	89.96±1.47	90.12±2.75	89.06±3.01	77.20±0.26	76.17±0.69	78.07±0.68	71.44±1.27	69.10±1.27	70.50±0.68	3.23±0.16	4.38±0.19	11.34±0.11
– same k_m	93.75±0.94	93.30±0.93	93.36±0.97	77.18±0.54	76.07±0.79	76.28±0.97	71.04±0.97	70.23±0.75	73.10±0.91	2.85±0.04	4.10±0.09	30.34±5.42

6.1 Ablations

We perform ablation experiments to study the main components of our method. First, we replace the multitaper tokenizer with a single view (single-kernel), reducing the method to a convolutional

Table 3: Performance comparison of tokenization methods for CVD and sleep using ECG and EEG.

Method	PhysioNet 2017			PTB-XL			Chapman			Sleep		
	Acc ↑	F1 ↑	W-F1 ↑	AUROC ↑	F1 ↑	Acc ↑	Acc ↑	F1 ↑	W-F1 ↑	κ ↑	F1 ↑	W-F1 ↑
Linear	68.03±0.29	50.54±1.40	65.92±0.66	91.10±0.31	70.20±0.55	60.94±1.15	96.88±0.20	89.28±1.11	96.83±0.19	71.75±0.51	63.75±0.22	85.61±0.14
Linear (ov.)	70.50±1.11	56.86±2.97	68.29±1.70	**91.39**±0.22	**71.18**±0.43	**60.91**±0.73	97.55±0.14	91.10±0.17	97.54±0.14	73.04±0.71	65.03±0.50	84.65±0.32
PathFormer	59.20±0.03	18.79±0.27	44.25±0.31	—	—	—	70.12±2.57	30.21±5.47	60.07±7.51	—	—	—
MedFormer	68.35±1.01	55.11±1.83	66.96±0.89	90.71±0.08	70.27±0.19	60.27±0.71	95.40±0.38	85.53±0.23	94.48±0.41	72.13±0.52	64.20±0.30	86.03±0.17
Totem	61.94±1.01	41.65±2.23	56.77±0.69	—	—	—	96.93±0.13	77.72±0.94	96.50±0.14	—	—	—
PerimidFormer	60.12±0.03	18.84±0.35	44.30±0.39	82.54±0.58	55.74±2.37	44.38±2.47	84.75±3.10	49.38±6.75	84.63±3.90	62.19±1.25	52.84±2.22	82.02±0.70
ScaleFormer	59.16±0.06	18.59±0.01	44.01±0.03	90.59±0.37	69.27±1.33	60.52±0.71	94.65±0.80	77.16±2.27	94.25±0.97	67.64±0.14	56.07±0.33	84.25±0.12
MTST	62.11±1.81	29.36±1.53	48.24±0.06	83.41±0.49	58.71±1.30	56.01±1.62	96.41±0.26	71.64±1.13	95.54±0.33	66.17±0.20	55.28±0.96	82.04±0.59
Kairos	68.93±1.72	56.26±0.11	66.46±1.39	88.25±0.45	66.38±1.03	57.04±0.86	97.94±0.08	91.36±0.35	97.98±0.08	73.06±0.65	63.55±0.76	86.22±0.39
Multitaper	**73.82**±0.96	**64.04**±1.60	**72.84**±0.68	90.76±0.03	69.14±0.37	60.04±0.97	**98.37**±0.39	**94.55**±1.08	**98.37**±0.39	**75.40**±0.30	**66.82**±0.34	**87.52**±0.09
– w/o weights	67.25±0.62	55.71±1.94	64.69±1.08	90.24±0.12	68.44±0.99	59.47±0.85	97.94±0.19	94.69±0.90	97.93±0.18	74.77±0.52	66.25±0.42	87.16±0.06
– single-kernel	71.16±1.42	61.58±2.05	69.73±1.81	89.76±0.47	66.40±2.04	59.13±0.64	95.96±0.46	85.71±1.61	95.96±0.46	66.86±1.74	56.48±0.52	84.47±0.45
– same k_m	71.86±0.50	62.30±0.85	70.37±0.92	90.10±0.21	67.60±0.39	59.16±1.30	97.18±0.07	94.04±0.32	97.59±0.22	73.78±0.62	66.05±0.17	86.75±0.13

tokenizer [50]. Second, we keep the multitaper but replace the learned weights with uniform sum (w/o weights). Third, we use the same number of samples in all views, as in traditional multitaper methods, instead of using distinct k_m across views (same k_m).

The ablations show that each component contributes to performance, with the multitaper providing the largest gain, while learned weighting and heterogeneous windows give additional improvements. An exception is UCIHAR, where using same k_m outperforms our method. One possible explanation is that introducing non-optimal multiple kernel sizes can increase the tradeoff. While the gates can in principle suppress less useful views, doing so reliably can be harder in smaller-data settings. Additional ablations and robustness experiments of our method, including sensitivity to embedding dimension, filter lengths k_m, and the tokenizer parameterization, are reported in Appendix E.

6.2 Windowing on Synthetic Tasks

To probe the approximation–estimation tradeoff under controlled conditions, we evaluate frequency, phase, and waveform-family classification across window lengths and noise levels, keeping the Transformer backbone fixed. We also compare two sampling regimes. Exp A uses sampling rate f_s and window length M_Δ, whereas Exp B doubles both, increasing sample support at the same duration $\Delta = M_\Delta / f_s$.

Figure 5 shows that the optimal token length varies across tasks and noise levels. Frequency classification favors intermediate windows, whereas waveform classification favors shorter windows that preserve local shape. Phase classification exhibits an intermediate pattern, with preferred windows shifting with SNR. These observations are consistent with the approximation–estimation interpretation. Full task definitions, training details, and results across model sizes appear in Appendix J.

6.3 Discussion

Our empirical results highlight several nuanced dynamics of temporal sequence tokenization, particularly regarding how different methods perform in the presence of noise for several modalities.

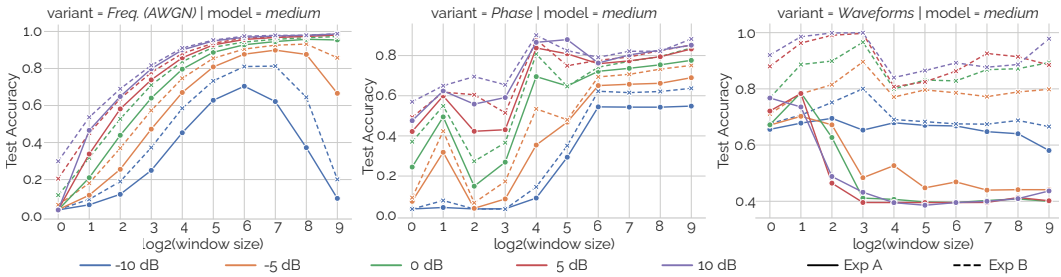

Figure 5: Window-size sensitivity on synthetic tasks. Test accuracy versus $\log_2 M_\Delta$ for frequency, phase, and waveform classification. Colors indicate SNR. At each horizontal position, Exp B uses twice the sampling rate and window length of Exp A, preserving temporal duration.

First, we observe that overlapping windows for tokenization frequently improve performance. For a fixed window length, overlap provides more window estimates without reducing the number of samples available to each estimate. The benefit of combining them depends on how strongly their errors are correlated. Overlapping windows share samples, so their residuals can be correlated even when the underlying sample noise is independent. The few cases of slight degradation arise on DaLiA and WildPPG, where noise such as motion artifacts [26] or temperature fluctuations [28] can be correlated across windows. In such settings, the conditional uncorrelatedness of noise between windows is weaker, so the gain can be reduced.

These empirical observations suggest that the proposed theoretical arguments remain informative even though our theory uses the segment-level representation as an average of window statistics for tractable analysis. Although this is a simplified surrogate of attention-based aggregation, the empirical results remain consistent with the theoretical predictions.

Furthermore, our experiments show that the frequency-driven tokenization strategies, such as PerimidFormer, perform worse for some datasets. These architectures utilize the Fast Fourier Transform (FFT) as a tool for defining their patching strategy. While this approach is intuitive for periodic signals, in regimes with low SNR, FFTs exhibit spurious peaks driven by noise [26, 60]. Consequently, methods that use FFT for windowing risk fitting their tokenization to the noise profile.

MedFormer and MTST provide an informative comparison for multiscale tokenization. Their multiscale construction introduces redundancy, since longer patches can contain the information captured by shorter ones, and cross-scale attention or concatenation integrate these views. However, because all scales are kept, it is possible that noisy or inaccurate estimations at some scales also affect the final representation. This suggests that redundancy can be most effective with a mechanism that selectively combines views, rather than passing all scales forward. In contrast, our method uses gates to sum views before the Transformer, providing a more controlled and efficient use of redundancy.

TOTEM, a codebook-based tokenizer trained with a reconstruction objective, suggests that preserving information for reconstruction is not by itself sufficient for downstream tasks. Although such objectives reduce approximation error, their weaker results indicate that reconstruction can be misaligned with representation learning, especially in noisy settings. A similar pattern appears for reconstruction-based autoencoders without masking [61, 62], which are not competitive. Overall, these results suggest that tokenizers should prioritize task-relevant structure over exact signal recovery.

Kairos is an adaptive-patching baseline because it does not rely on a single size. Instead, it uses coarse segmentation, routes segments to a subset of granularity experts, and fuses their embeddings through learned weights. Its performance therefore depends not only on tokenization, but also on the routing and fusion mechanisms. By contrast, our method modifies tokenization directly by improving estimates within the original pipeline, without introducing routing modules or separate objectives.

Overall, we conjecture that part of the challenge is the effective dimension of the window-to-token map. Although the Transformer receives a fixed d-dimensional token, the tokenizer must compress a larger input space as the window length increases, from $C_{\text{in}}M_{\Delta}$ samples into the same token dimension. For estimation on such windows, this problem can become much harder as the window length increases, with sample complexity growing rapidly, potentially exponentially, with the effective dimension [63]. Thus, while longer windows increase temporal support, they also enlarge the function space or statistical features that must be estimated from each window.

Due to space constraints, we include additional experiments and discussions in Appendix; see Section 9. In particular, Appendix F presents self-supervised learning (SSL) experiments. Appendix H gives computational comparison of methods. Appendix J presents controlled experiments with synthetic data that further support the connection between the theory and the empirical observations.

7 Related Work

Investigating tokenization. Tokenization is a core design choice across domains. In language, it has been dominated by subword methods such as Byte-Pair Encoding (BPE) [64, 65] and WordPiece [66], while in vision the rise of Transformers made patching the standard tokenization strategy [67]. Recent work has examined tokenization itself as a source of inductive bias rather than a purely preprocessing decision [68]. For instance, in language modeling, tokenization can help capture higher-order Markov

structure more effectively than character-level representations [69], and vocabulary and segmentation choices can systematically change probabilities and sequence-length preferences [2].

Our setting differs in an important way from the literature. Language is commonly modeled as a sequence of symbols [9, 70, 71], and many such models idealize the input as error-free symbolic sequence rather than a noisy continuous observation. By contrast, signals are sampled physical measurements with finite resolution and SNR [72]. A temporal token must summarize a window of noisy samples. Thus, unlike prior work, we focus on how tokenization in temporal data induces a tradeoff through the interaction of compression and finite-sample estimation.

Closer to ours, recent work studies patching as a source of bias, arguing that the patching affects temporal structure, and that larger patches favor low-frequency [73]. This perspective is related to ours as it argues tokenization shapes behavior. However, our theoretical and empirical investigation suggests a more nuanced picture. Larger patches do not categorically prevent the model from extracting high-frequencies, since gradient-based learning can still recover task-relevant high-frequencies within a long window. Rather, increasing the window length makes tokenization a harder compression problem [74], because a token must summarize a longer window. In this regime, the model can still capture high-frequency when it is task-relevant (see Appendix J.6), but the approximation burden increases, making it difficult to preserve all within-window variation uniformly.

Temporal sequence tokenization. A large and growing body of literature has proposed different tokenization methods. Early models relied on point-wise projections by aggressively discretizing continuous values into finite text-like vocabularies [75], but struggled with high-frequency noise and lacked local semantic context [76]. To overcome this, patching was introduced to aggregate local time steps into a single token [23, 76]. While patching improves performance by establishing a local field and reducing the length, it relies on projections of raw samples, leaving the model sensitive to the hyperparameter of window size [77]. Parallel lines of work attempt to bypass temporal aggregation entirely by treating entire variates as independent tokens (e.g., iTransformer [78]), or adaptively segmenting the signal [57, 79]. However, these methods either ignore local estimation or require estimating the information density of each window through an additional optimization procedure, which is challenging in practice [68, 80–83] and often introduces many hyperparameters [84]. By contrast, we formalize the tradeoff and introduce a tokenization that avoids a separate objective.

8 Limitations

Our analysis models the segment representation as a weighted sum of window statistics, with the weights held fixed. It includes unequal weighting and correlated estimation errors across windows. In learned attention, the weights are computed from the tokens themselves, so noise in a window can change both its representation and how much attention it receives. Our calculation accounts for errors in the window representations but does not track the resulting changes in attention weights.

At the same time, this abstraction is consistent with the information bottleneck view of learning [33], which seeks representations that preserve task-relevant information after a mapping function. It is also aligned with the data processing inequality [35], since any aggregation mechanism operates on the tokens and therefore cannot create task-relevant information that was discarded during tokenization. In this sense, our approach is principled even though simplified. A direction for future work can be to investigate when/if attention can recover features that are weakened, but not removed, by limited sample support or compression, and when such information is irreversibly lost during tokenization.

9 Conclusion

We demonstrate that converting time series into tokens is not a formatting step, but a fundamental tradeoff between approximation and estimation. Building on this, we propose a lightweight method that reduces the burden of window-size tuning while leaving the Transformer architecture unchanged. Despite these improvements, the underlying challenge of temporal sequence modeling remains. Real-world signal dynamics are heterogeneous; therefore, we argue that the ideal temporal tokenizer may need to move beyond fixed partitioning. Instead, it should act as an information-theoretic bottleneck, dynamically adjusting its windows based on the signal's content, noise, and information density. Developing such mechanisms is an important next step toward capable temporal models.

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[2] Pietro Lesci, Clara Meister, Thomas Hofmann, Andreas Vlachos, and Tiago Pimentel. Causal estimation of tokenisation bias. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar, editors, *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vienna, Austria, July 2025. Association for Computational Linguistics.

[3] Jose Camacho-Collados and Mohammad Taher Pilehvar. From word to sense embeddings: a survey on vector representations of meaning. *J. Artif. Int. Res.*, 63(1):743–788, September 2018.

[4] Jonathan J. Webster and Chunyu Kit. Tokenization as the initial phase in nlp. In *Proceedings of the 14th Conference on Computational Linguistics - Volume 4*, COLING '92, page 1106–1110, USA, 1992. Association for Computational Linguistics.

[5] David A. Haslett. Tokenization changes meaning in large language models: Evidence from Chinese. *Computational Linguistics*, 51(3):785–814, September 2025.

[6] Nitin Indurkhya and Fred J. Damerau. *Handbook of Natural Language Processing*. Chapman & Hall/CRC, 2nd edition, 2010.

[7] Claire Kramsch. *Introduction*. Key Topics in Applied Linguistics. Cambridge University Press, 2020.

[8] Ferdinand de Saussure. *Course in General Linguistics*. McGraw-Hill, New York, 1966.

[9] Jerry A. Fodor. *The Language of Thought*. Harvard University Press, 1975.

[10] Stefan L. Frank, Rens Bod, and Morten H. Christiansen. How hierarchical is language use? *Proceedings of the Royal Society B: Biological Sciences*, 279(1747):4522–4531, 09 2012.

[11] François Pellegrino, Christophe Coupé, and Egidio Marsico. Across-language perspective on speech information rate. *Language*, 87:539 – 558, 2011.

[12] Ruben Cartuyvels, Graham Spinks, and Marie-Francine Moens. Discrete and continuous representations and processing in deep learning: Looking forward. *AI Open*, 2:143–159, 2021.

[13] Ayush Kaushal and Kyle Mahowald. What do tokens know about their characters and how do they know it? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2487–2507, Seattle, United States, July 2022. Association for Computational Linguistics.

[14] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*. 3rd edition, 2026. Online manuscript released January 6, 2026.

[15] Christopher D. Manning and Hinrich Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, Cambridge, MA, May 1999.

[16] Lorenzo Ferrone and Fabio Massimo Zanzotto. Symbolic, distributed, and distributional representations for natural language processing in the era of deep learning: A survey. *Frontiers in Robotics and AI*, 2020.

[17] Edward Gibson, Richard Futrell, Steven P. Piantadosi, Isabelle Dautriche, Kyle Mahowald, Leon Bergen, and Roger Levy. How efficiency shapes human language. *Trends in Cognitive Sciences*, 23(5):389–407, 2019.

[18] Saima Malik-Moraleda, Dima Ayyash, Jeanne Gallée, Josef Affourtit, Malte Hoffmann, Zachary Mineroff, Olessia Jouravlev, and Evelina Fedorenko. An investigation across 45 languages and 12 language families reveals a universal language network. *Nature Neuroscience*, 25(8):1014–1019, August 2022.

[19] Alan V. Oppenheim, Alan S. Willsky, and S. Hamid Nawab. *Signals & Systems (2nd Ed.)*. Prentice-Hall, Inc., USA, 1996.

[20] Simon Haykin and Barry Van Veen. *Signals and Systems*. John Wiley & Sons, Inc., USA, 2nd edition, 2002.

[21] Yuxuan Wang, Haixu Wu, Jiaxiang Dong, Guo Qin, Haoran Zhang, Yong Liu, Yunzhong Qiu, Jianmin Wang, and Mingsheng Long. Timexer: Empowering transformers for time series forecasting with exogenous variables. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[22] Simo Särkkä and Lennart Svensson. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press, Cambridge, 2 edition, 2023.

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

[24] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: a framework for self-supervised learning of speech representations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[25] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, Cambridge, 2008.

[26] Berken Utku Demirel and Christian Holz. An unsupervised approach for periodic source detection in time series. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

[27] Hangwei Qian, Tian Tian, and Chunyan Miao. What makes good contrastive learning on small-scale wearable-based tasks? In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 3761–3771, New York, NY, USA, 2022. Association for Computing Machinery.

[28] Manuel Meier and et al. WildPPG: A real-world PPG dataset of long continuous recordings. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.

[29] Larry Wasserman. *All of Statistics: A Concise Course in Statistical Inference*. Springer Publishing Company, Incorporated, 2010.

[30] E. L. Lehmann and George Casella. *Theory of Point Estimation*. Springer Texts in Statistics. Springer-Verlag, New York, 1998.

[31] John Rice. Bandwidth choice for nonparametric regression. *The Annals of Statistics*, 12(4):1215–1230, 1984.

[32] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[33] Naftali Tishby, Fernando C Pereira, and William Bialek. The information bottleneck method. *ArXiv*, physics/0004057, 2000.

[34] D. Rife and R. Boorstyn. Single tone parameter estimation from discrete-time observations. *IEEE Transactions on Information Theory*, 20(5):591–598, 1974.

[35] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[36] Marco A. F. Pimentel, Alistair E. W. Johnson, Peter H. Charlton, Drew Birrenkott, Peter J. Watkinson, Lionel Tarassenko, and David A. Clifton. Toward a robust estimation of respiratory rate from pulse oximeters. *IEEE Transactions on Biomedical Engineering*, 64(8):1914–1923, 2017.

[37] D. J. Thomson. Spectrum estimation and harmonic analysis. *Proceedings of the IEEE*, 70(9):1055–1096, 1982.

[38] Zhilin Zhang, Zhouyue Pi, and Benyuan Liu. Troika: A general framework for heart rate monitoring using wrist-type photoplethysmographic signals during intensive physical exercise. *IEEE Transactions on Biomedical Engineering*, 62(2):522–531, 2015.

[39] Attila Reiss, Ina Indlekofer, Philip Schmidt, and Kristof Van Laerhoven. Deep ppg: Large-scale heart rate estimation with convolutional neural networks. *Sensors*, 19, 2019.

[40] D. Anguita, Alessandro Ghio, L. Oneto, Xavier Parra, and Jorge Luis Reyes-Ortiz. Human activity recognition on smartphones using a multiclass hardware-friendly support vector machine. In *International Workshop on Ambient Assisted Living and Home Care*, 2012.

[41] Allan Stisen, Henrik Blunck, Sourav Bhattacharya, Thor Siiger Prentow, Mikkel Baun Kjærgaard, Anind Dey, Tobias Sonne, and Mads Møller Jensen. Smart devices are different: Assessing and mitigatingmobile sensing heterogeneities for activity recognition. SenSys, 2015.

[42] Mi Zhang and Alexander A. Sawchuk. Usc-had: A daily activity dataset for ubiquitous activity recognition using wearable sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*. Association for Computing Machinery, 2012.

[43] Ryan Mattfeld, Elliot Jesch, and Adam Hoover. A new dataset for evaluating pedometer performance. In *2017 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 865–869, 2017.

[44] Gari D Clifford, Chengyu Liu, Benjamin Moody, Li-wei H. Lehman, Ikaro Silva, Qiao Li, A E Johnson, and Roger G. Mark. Af classification from a short single lead ecg recording: The physionet/computing in cardiology challenge 2017. In *2017 Computing in Cardiology (CinC)*, pages 1–4, 2017.

[45] Patrick Wagner, Nils Strodthoff, Ralf-Dieter Bousseljot, Dieter Kreiseler, Fatima I. Lunze, Wojciech Samek, and Tobias Schaeffter. PTB-XL, a large publicly available electrocardiography dataset. *Scientific Data*, 7(1):154, May 2020.

[46] Jianwei Zheng, Jianming Zhang, Sidy Danioko, Hai Yao, Hangyuan Guo, and Cyril Rakovski. A 12-lead electrocardiogram database for arrhythmia research covering more than 10,000 patients. *Scientific Data*, 7(1):48, February 2020.

[47] B. Kemp, A.H. Zwinderman, B. Tuk, H.A.C. Kamphuisen, and J.J.L. Oberye. Analysis of a sleep-dependent neuronal feedback loop: the slow-wave microcontinuity of the eeg. *IEEE Transactions on Biomedical Engineering*, 47(9):1185–1194, 2000.

[48] Ary L. Goldberger, Luis A. N. Amaral, Leon Glass, Jeffrey M. Hausdorff, Plamen Ch. Ivanov, Roger G. Mark, Joseph E. Mietus, George B. Moody, Chung-Kang Peng, and H. Eugene Stanley. Physiobank, physiotoolkit, and physionet. *Circulation*, 101(23):e215–e220, 2000.

[49] Guagnyu Wang, Wenchao Liu, Yuhong He, Cong Xu, Lin Ma, and Haifeng Li. Eegpt: Pretrained transformer for universal and reliable representation of eeg signals. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 39249–39280. Curran Associates, Inc., 2024.

[50] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 22–31, 2021.

[51] Sabera J Talukder, Yisong Yue, and Georgia Gkioxari. TOTEM: TOkenized time series EMbeddings for general time series analysis. *Transactions on Machine Learning Research*, 2024.

[52] Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. Neural discrete representation learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[53] Peng Chen, Yingying ZHANG, Yunyao Cheng, Yang Shu, Yihang Wang, Qingsong Wen, Bin Yang, and Chenjuan Guo. Pathformer: Multi-scale transformers with adaptive pathways for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024.

[54] Yihe Wang, Nan Huang, Taida Li, Yujun Yan, and Xiang Zhang. Medformer: A multi-granularity patching transformer for medical time-series classification. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[55] Mohammad Amin Shabani, Amir H. Abdi, Lili Meng, and Tristan Sylvain. Scaleformer: Iterative multi-scale refining transformers for time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

[56] Yitian Zhang, Liheng Ma, Soumyasundar Pal, Yingxue Zhang, and Mark Coates. Multi-resolution time-series transformer for long-term forecasting. In Sanjoy Dasgupta, Stephan Mandt, and Yingzhen Li, editors, *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 4222–4230. PMLR, 02–04 May 2024.

[57] Kun Feng, Shaocheng Lan, Yuchen Fang, Wenchao He, Lintao Ma, Xingyu Lu, and Kan Ren. Kairos: Towards adaptive and generalizable time series foundation models. *arXiv preprint arXiv:2509.25826*, 2025.

[58] Qiang Wu, Gechang Yao, Zhixi Feng, and Shuyuan Yang. Peri-midformer: Periodic pyramid transformer for time series analysis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[59] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[60] Petre Stoica and Randolph L. Moses. *Spectral Analysis of Signals*. Pearson Prentice Hall, 2005. Google-Books-ID: h78ZAQAAIAAJ.

[61] Michael Tschannen, Olivier Bachem, and Mario Lucic. Recent advances in autoencoder-based representation learning. In *Third workshop on Bayesian Deep Learning (NeurIPS 2018)*, 2018.

[62] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[63] Zahra Kadkhodaie, Florentin Guth, Eero P Simoncelli, and Stéphane Mallat. Generalization in diffusion models arises from geometry-adaptive harmonic representations. In *The Twelfth International Conference on Learning Representations*, 2024.

[64] Philip Gage. A new algorithm for data compression. *C Users J.*, 12(2):23–38, February 1994.

[65] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[66] Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5149–5152, 2012.

[67] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[68] Marius Aasan, Martine Hjelkrem-Tan, Nico Catalano, Changkyu Choi, and Adín Ramírez Rivera. Differentiable hierarchical visual tokenization. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2026.

[69] Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. An analysis of tokenization: Transformers under markov data. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, editors, *Advances in Neural Information Processing Systems*, volume 37, pages 62503–62556. Curran Associates, Inc., 2024.

[70] Richard Futrell and Michael Hahn. Linguistic structure from a bottleneck on sequential information processing. *Nature Human Behaviour*, 10(3):589–600, March 2026.

[71] Serhii Havrylov and Ivan Titov. Emergence of language with multi-agent games: learning to communicate with sequences of symbols. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 2146–2156, Red Hook, NY, USA, 2017. Curran Associates Inc.

[72] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice Hall Press, USA, 3rd edition, 2009.

[73] Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Bernie Wang. Understanding the implicit biases of design choices for time series foundation models. In *The Fourteenth International Conference on Learning Representations*, 2026.

[74] Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. Wiley, 1 edition, October 2001.

[75] Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Transactions on Machine Learning Research*, 2024. Expert Certification.

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

[77] Donghao Luo and Xue Wang. DeformableTST: Transformer for time series forecasting without over-reliance on patching. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

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

[79] Jianyang Qin, Chaoyang Li, Jinhao Cui, Lingzhi Wang, Zhao Liu, and Qing Liao. Bridging time and linguistics: LLMs as time series analyzer through symbolization and segmentation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[80] Kai Wei, Rishabh Iyer, and Jeff Bilmes. Fast multi-stage submodular maximization. In Eric P. Xing and Tony Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1494–1502, Bejing, China, 22–24 Jun 2014. PMLR.

[81] Andreas Krause, Ajit Singh, and Carlos Guestrin. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *Journal of Machine Learning Research*, 9(8):235–284, 2008.

[82] Gavin Brown, Adam Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *Journal of Machine Learning Research*, 13(2):27–66, 2012.

[83] Sachith Abeywickrama, Emadeldeen Eldele, Min Wu, Xiaoli Li, and Chau Yuen. EntroPE: Entropy-guided dynamic patch encoder for time series forecasting, 2026.

[84] Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, Laszlo A. Jeni, and Kris Kitani. Faster vision transformers with adaptive patches. In *The Fourteenth International Conference on Learning Representations*, 2026.

[85] Çağatay Candan and Utku Çelebi. Frequency estimation of a single real-valued sinusoid: An invariant function approach. *Signal Processing*, 185:108098, 2021.

[86] Charles J. Stone. Optimal Global Rates of Convergence for Nonparametric Regression. *The Annals of Statistics*, 10(4):1040–1053, December 1982.

[87] Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Series in Statistics. Springer, New York, NY, 2009.

[88] D. Rife and R. Boorstyn. Single tone parameter estimation from discrete-time observations. *IEEE Transactions on Information Theory*, 20(5):591–598, 1974.

[89] Duc Son Pham and Abdelhak M. Zoubir. Analysis of multicomponent polynomial phase signals. *IEEE Transactions on Signal Processing*, 55(1):56–65, 2007.

[90] P.M. Djuric and S.M. Kay. Parameter estimation of chirp signals. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(12):2118–2126, 1990.

[91] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery.

[92] Yin Tang, Lei Zhang, Qi Teng, Fuhong Min, and Aiguo Song. Triple cross-domain attention on human activity recognition using wearable sensors. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 6(5):1167–1176, 2022.

[93] Congcong Ma, Wenfeng Li, Jingjing Cao, Juan Du, Qimeng Li, and Raffaele Gravina. Adaptive sliding window based activity recognition for assisted livings. *Information Fusion*, 53:55–65, 2020.

[94] Zhiguo Zhang. Spectral and Time-Frequency Analysis. In Li Hu and Zhiguo Zhang, editors, *EEG Signal Processing and Feature Extraction*, pages 89–116. Springer, Singapore, 2019.

[95] D. Slepian. On bandwidth. *Proceedings of the IEEE*, 64(3):292–300, 1976.

[96] 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 Min. Knowl. Discov.*, 34(6):1936–1962, November 2020.

[97] Hui Tian. *Noise analysis in CMOS image sensors*. PhD thesis, United States – California, 2000. Ph.D.

[98] B. Widrow. Statistical analysis of amplitude-quantized sampled-data systems. *Transactions of the American Institute of Electrical Engineers, Part II: Applications and Industry*, 79(6):555–568, 1961.

[99] Bernard Widrow and István Kollár. *Quantization Noise: Roundoff Error in Digital Computation, Signal Processing, Control, and Communications*. Cambridge University Press, USA, 2008.

[100] Zaid J. Towfic, Shang-Kee Ting, and Ali H. Sayed. Sampling clock jitter estimation and compensation in adc circuits. In *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 829–832, 2010.

[101] Jo Woon Chong, Duy K. Dao, S. M. A. Salehizadeh, David D. McManus, Chad E. Darling, Ki H. Chon, and Yitzhak Mendelson. Photoplethysmograph Signal Reconstruction Based on a Novel Hybrid Motion Artifact Detection–Reduction Approach. Part I: Motion and Noise Artifact Detection. *Annals of Biomedical Engineering*, 42(11):2238–2250, November 2014.

[102] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic gradient descent as approximate bayesian inference. *J. Mach. Learn. Res.*, 18(1):4873–4907, January 2017.

[103] Gaetano Romano, Guillem Rigaill, Vincent Runge, and Paul Fearnhead. Detecting Abrupt Changes in the Presence of Local Fluctuations and Autocorrelated Noise. *Journal of the American Statistical Association*, 117(540):2147–2162, October 2022.

[104] Jathurshan Pradeepkumar, Xihao Piao, Zheng Chen, and Jimeng Sun. Tokenizing single-channel EEG with time-frequency motif learning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[105] Leandro Giacomini Rocha, Dwaipayan Biswas, Bram-Ernst Verhoef, Sergio Bampi, Chris Van Hoof, Mario Konijnenburg, Marian Verhelst, and Nick Van Helleputte. Binary cornet: Accelerator for hr estimation from wrist-ppg. *IEEE Transactions on Biomedical Circuits and Systems*, 2020.

[106] Mohammed Saeed, Mauricio Villarroel, Andrew T. Reisner, Gari Clifford, Li-Wei Lehman, George Moody, Thomas Heldt, Tin H. Kyaw, Benjamin Moody, and Roger G. Mark. Multiparameter intelligent monitoring in intensive care ii: A public-access intensive care unit database. *Critical Care Medicine*, 39(5), 2011.

[107] Dwaipayan Biswas, Luke Everson, Muqing Liu, Madhuri Panwar, Bram-Ernst Verhoef, Shrishail Patki, Chris H. Kim, Amit Acharyya, Chris Van Hoof, Mario Konijnenburg, and Nick Van Helleputte. Cornet: Deep learning framework for ppg-based heart rate estimation and biometric identification in ambulant environment. *IEEE Transactions on Biomedical Circuits and Systems*, 2019.

[108] Hangwei Qian, Sinno Jialin Pan, and Chunyan Miao. Latent independent excitation for generalizable sensor-based cross-person activity recognition. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11921–11929, May 2021.

[109] Berken Utku Demirel and Christian Holz. Finding order in chaos: A novel data augmentation method for time series in contrastive learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[110] Annan Yu, Danielle C. Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W. Mahoney, and Bernie Wang. Understanding transformers for time series: Rank structure, flow-of-ranks, and compressibility. In *The Fourteenth International Conference on Learning Representations*, 2026.

[111] 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, and Bernie Wang. Enhancing foundation models for time series forecasting via wavelet-based tokenization. In *Forty-second International Conference on Machine Learning*, 2025.

[112] Stéphane Mallat. *A Wavelet Tour of Signal Processing, Third Edition: The Sparse Way*. Academic Press, Inc., USA, 3rd edition, 2008.

[113] Tai Sing Lee. Image representation using 2d gabor wavelets. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(10):959–971, 1996.

[114] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023.

[115] Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, and Qingsong Wen. Time-LLM: Time series forecasting by reprogramming large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[116] Chengsen Wang, Qi Qi, Jingyu Wang, Haifeng Sun, Zirui Zhuang, Jinming Wu, Lei Zhang, and Jianxin Liao. Chattime: a unified multimodal time series foundation model bridging numerical and textual data. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'25/IAAI'25/EAAI'25. AAAI Press, 2025.

[117] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

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

[119] Berken Utku Demirel and Christian Holz. Learning without augmenting: Unsupervised time series representation learning via frame projections. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[120] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[121] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H. Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, Bilal Piot, Koray Kavukcuoglu, Rémi Munos, and Michal Valko. Bootstrap your own latent: a new approach to self-supervised learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[122] Zhihan Yue, Yujing Wang, Juanyong Duan, Tianmeng Yang, Congrui Huang, Yunhai Tong, and Bixiong Xu. Ts2vec: Towards universal representation of time series. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8980–8987, Jun. 2022.

[123] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[124] Emadeldeen Eldele, Mohamed Ragab, Zhenghua Chen, Min Wu, Chee Keong Kwoh, Xiaoli Li, and Cuntai Guan. Time-series representation learning via temporal and contextual contrasting. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 2021.

[125] Xiaowen Ma, Zhen-Liang Ni, Shuai Xiao, and Xinghao Chen. Timepro: Efficient multivariate long-term time series forecasting with variable- and time-aware hyper-state. In *Forty-second International Conference on Machine Learning*, 2025.

[126] Zihan Wang, Fanheng Kong, Shi Feng, Ming Wang, Xiaocui Yang, Han Zhao, Daling Wang, and Yifei Zhang. Is mamba effective for time series forecasting? *Neurocomputing*, 619:129178, 2025.

Appendix Overview

A Theorems **20**

A.1 Notation and preliminaries 20

A.2 Propositions 21

A.2.1 Bounding the bias 21

A.2.2 Bounding the variance 22

A.3 Examples of error sources 27

A.4 Overlapping windows 28

B Datasets **30**

C Baseline Details **32**

C.1 Only-Tokenization 32

C.2 Joint tokenization–architecture 32

C.3 Further discussion of related work 33

D Implementation Details **34**

D.1 Multitaper algorithm 34

D.2 Architecture 35

D.3 Baselines 35

E Robustness and Significance Analysis **38**

E.1 Gating weights 40

E.2 Significance analysis 40

F Self-Supervised Learning Experiments **42**

G Time-Series Forecasting **43**

H Computational Overhead of Techniques **44**

I Motivating Example **47**

I.1 Further evidence with other tasks 47

I.2 Revisiting the motivational example 49

I.3 Empirical fit 49

J Further Experiments with Synthetic Data **51**

J.1 Protocol 51

J.2 Tasks and theory 52

J.3 Effect of windowing across tasks 53

J.4 Windowing and entropy 54

J.5 Temporal versus sample support 55

J.6 Differentiating low and high frequency 56

J.7 Noise type investigation 58

J.8 Dimension of the space 59

J.9 Training and scaling 60

J.10 Connection back to theory 60

A Theoretical Analysis

Here, we present complete proofs of our theoretical study, starting with notations.

A.1 Representations and Notations

Table 4: Main notations

Notation	Description
$\mathbf{x}$	Input temporal sequence, $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times T f_s}$.
$\mathbf{X}$	Generic matrix-valued quantity, such as a weight matrix.
f_s	Sampling rate in Hz.
T	Sequence duration, in seconds.
Δ	Window (patch) length, in seconds.
M_Δ	Number of samples in one window, given by $M_\Delta = f_s\Delta \in \mathbb{N}$.
N	Number of nonoverlapping windows, $N = T/\Delta$.
ω_i	Fixed aggregation weight, with $\omega_i \geq 0$ and $\sum_i \omega_i = 1$.
$N_{\text{eff}}(\Delta)$	Effective number of contributing windows, $N_{\text{eff}}(\Delta) = (\sum_i \omega_i^2)^{-1}$.
C_{in}	Number of input channels.
I_i	The i-th temporal window interval.
$\mathbf{w}_i$	The i-th window extracted from the sequence, defined by $\mathbf{w}_i = \mathbf{x}\|_{I_i}$.
$x_i[m]$	Scalar observation at sample index m within window i.
$\mathcal{W}_\Delta$	Space of windows of duration Δ.
$\hat{\phi}_\Delta$	Learned tokenizer map from a window to a d-dimensional token.
$\phi^\star_\Delta$	Oracle map defining the ideal target feature of a window.
$\mathbf{z}_i$	Token representation of window $\mathbf{w}_i$, i.e., $\mathbf{z}_i = \hat{\phi}_\Delta(\mathbf{w}_i)$.
$\mathbf{Z}$	Sequence of tokens, $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{N \times d}$.
$\theta^\star_i$	Target statistic of window i, defined by $\theta^\star_i = \phi^\star_\Delta(\mathbf{w}_i)$.
$\hat{\theta}_i$	Empirical statistic of window i, defined by $\hat{\theta}_i = \hat{\phi}_\Delta(\mathbf{w}_i)$.
$\theta^\star$	Segment-level target statistic, $\theta^\star = \sum_{i=1}^N \omega_i \theta^\star_i$.
$\hat{\theta}$	Segment-level empirical statistic, $\hat{\theta} = \sum_{i=1}^N \omega_i \hat{\theta}_i$.
b_i	Deterministic error component on window i.
ξ_i	Random error component on window i.
$\bar{b}$	Weighted bias, $\bar{b} = \sum_{i=1}^N \omega_i b_i$.
$\bar{\xi}$	Weighted residual, $\bar{\xi} = \sum_{i=1}^N \omega_i \xi_i$.
T_Δ	Token-induced approximation operator acting on a window.
$\mathcal{S}_\Delta$	Class of surrogate windows representable by one token at scale Δ.
$\tilde{\theta}_i$	Intermediate target statistic, defined by $\tilde{\theta}_i = \phi^\star_\Delta(T_\Delta \mathbf{w}_i)$.
L_ϕ	Lipschitz constant of the oracle map $\phi^\star_\Delta$.
C_{app}	Approximation constant controlling how well $T_\Delta \mathbf{w}$ approximates $\mathbf{w}$.
r	Approximation exponent in the bias bound.
c	Constant in the single-window variance law.
p	Estimation exponent controlling how the window-level variance scales.
a	Variance constant in the duration domain, $a = c f_s^{-p}$.
K	Number of multitaper estimates.
$\|\mathbf{x}\|$	Vector norm.

A.2 Propositions

In this section, we give the detailed proofs for our theoretical analysis.

Centered-error convention and token-induced bias. The decomposition in Equation 1 follows the standard bias–variance convention. All mean error is assigned to the systematic term, and the residual is defined to be centered. Thus, b_i denotes the systematic component for window i, not only one specific error source.

Since our focus is tokenization, we isolate the token-induced component of the systematic error by defining $\tilde{\theta}_i := \phi^\star_\Delta(T_\Delta \mathbf{w}_i)$ and $r_i := \hat{\phi}_\Delta(\mathbf{w}_i) - \phi^\star_\Delta(T_\Delta \mathbf{w}_i)$. Then

$$\hat{\theta}_i - \theta^\star_i = \underbrace{\big(\phi^\star_\Delta(T_\Delta \mathbf{w}_i) - \phi^\star_\Delta(\mathbf{w}_i)\big)}_{b_i^{\mathrm{tok}}} + \underbrace{\mathbb{E}[r_i]}_{b_i^{\mathrm{learn}}} + \underbrace{\big(r_i - \mathbb{E}[r_i]\big)}_{\xi_i}$$

Here, b_i^{tok} is the systematic error caused by replacing the original window with its token surrogate, while b_i^{learn} captures any remaining mean error of the learned estimator around the surrogate target. By construction, the final term is centered, since $\mathbb{E}[\xi_i] = \mathbb{E}[r_i - \mathbb{E}[r_i]] = 0$. Therefore, there is no additional mean component left in ξ_i; any such component is already included in b_i^{learn}.

The main text uses the centered-estimator case, $\mathbb{E}[r_i] = 0$, so that $b_i^{\mathrm{learn}} = 0$ and the systematic term reduces to the token-induced component $b_i = b_i^{\mathrm{tok}}$. More generally, if $b_i^{\mathrm{learn}} \neq 0$, it is absorbed into the total bias $b_i = b_i^{\mathrm{tok}} + b_i^{\mathrm{learn}}$. This can shift the overall error level, but it is not an omitted variance term. Our window-size analysis isolates b_i^{tok}, because this is the component directly controlled by the tokenization scale Δ, which is the focus of this work.

A.2.1 Bounding the bias

To prove Proposition 3.1, we first formalize the approximation induced by tokenizing a window at scale Δ.

Definition A.1 (Quasi-optimal token-induced approximation)**.** Let $\mathcal{W}_\Delta$ be a window space, and $\mathcal{S}_\Delta \subset \mathcal{W}_\Delta$ denote the class of surrogate windows representable by one token at scale Δ. We say that $T_\Delta : \mathcal{W}_\Delta \to \mathcal{S}_\Delta$ is C_{qo}-quasi-optimal with respect to window norm $\|\cdot\|_{\mathcal{W}_\Delta}$ if

$$\|w - T_\Delta w\|_{\mathcal{W}_\Delta} \le C_{\mathrm{qo}} \inf_{s \in \mathcal{S}_\Delta} \|w - s\|_{\mathcal{W}_\Delta} \qquad \text{for all } w \in \mathcal{W}_\Delta,$$

for some constant $C_{\mathrm{qo}} \ge 1$ independent of w and Δ.

This definition is useful because the bias term depends on how well the tokenized surrogate $T_\Delta w$ approximates the original window w. The set $\mathcal{S}_\Delta$ is the class of window surrogates representable by one token at scale Δ. The quasi-optimality requires that this surrogate achieve, up to a constant factor C_{qo}, the best approximation error available within $\mathcal{S}_\Delta$. The definition itself does not assume that tokenization loses information. If $\mathcal{S}_\Delta$ is sufficiently expressive, then the infimum may be zero. Its role is to compare the approximation $T_\Delta w$ to the best surrogate by the tokenization class.

To turn this definition into the approximation estimate for Proposition 3.1, we combine optimality with an approximation bound for the surrogate class $\mathcal{S}_\Delta$, where the bound depends on the window.

Lemma A.2 (Approximation rate of the token-representable surrogate class)**.** *Let $T_\Delta : \mathcal{W}_\Delta \to \mathcal{S}_\Delta$ be C_{qo}-quasi-optimal with respect to $\|\cdot\|_{\mathcal{W}_\Delta}$. Assume that there exist a seminorm $|\cdot|_{\mathcal{X}_\Delta}$, constants $C_0 > 0$ and $r > 0$, such that*

$$\inf_{s \in \mathcal{S}_\Delta} \|w - s\|_{\mathcal{W}_\Delta} \le C_0 \Delta^r |w|_{\mathcal{X}_\Delta}, \quad \textit{where} \quad |w|_{\mathcal{X}_\Delta} \le M \qquad \textit{for all } w \in \mathcal{W}_\Delta,$$

for some constant $M > 0$. Then the token-induced approximation error satisfies

$$\|w - T_\Delta w\|_{\mathcal{W}_\Delta} \le C_{\mathrm{app}} \Delta^r \qquad \textit{for all } w \in \mathcal{W}_\Delta,$$

where $C_{\mathrm{app}} := C_{\mathrm{qo}} C_0 M$.

Proof. Fix $w \in \mathcal{W}_\Delta$. By quasi-optimality of T_Δ,

$$\|w - T_\Delta w\|_{\mathcal{W}_\Delta} \le C_{\mathrm{qo}} \inf_{s \in \mathcal{S}_\Delta} \|w - s\|_{\mathcal{W}_\Delta}$$

Applying the approximation estimate for $\mathcal{S}_\Delta$ gives

$$\|w - T_\Delta w\|_{\mathcal{W}_\Delta} \le C_{\mathrm{qo}} C_0 \Delta^r |w|_{\mathcal{X}_\Delta}$$

Using the uniform bound $|w|_{\mathcal{X}_\Delta} \le M$, we obtain

$$\|w - T_\Delta w\|_{\mathcal{W}_\Delta} \le C_{\mathrm{qo}} C_0 M \Delta^r$$

We can end the proof by setting $C_{\mathrm{app}} := C_{\mathrm{qo}} C_0 M$. □

In this Lemma, $\mathcal{S}_\Delta$ describes what one token can represent, $T_\Delta w$ is the surrogate actually chosen for the window w, C_{qo} measures how close that choice is to the best possible one in $\mathcal{S}_\Delta$, and r measures how the approximation improves as the window becomes shorter. Thus, the approximation rate is controlled jointly by the expressiveness of the representable class and the quality of the tokenization.

Remark A.3 (Window norm.). We intentionally left the norm $\|\cdot\|_{\mathcal{W}_\Delta}$ abstract. It may be chosen as L^2, L^∞, or an appropriate Sobolev norm, depending on the task and the surrogate family $\mathcal{S}_\Delta$. Since our inputs are finite-length sampled sequences, each window admits a natural discrete L^2 norm.

Since these tools and terms are abstract, we give two examples and the corresponding definitions.

Example 1: mean-retaining token. Suppose one token retains only the average value of the window. Then $\mathcal{S}_\Delta$ is the class of constant functions on the window, and $T_\Delta w$ is the constant function equal to the window mean. In $L^2([0,\Delta])$, this is the best constant approximation, so $C_{\mathrm{qo}} = 1$. If the windows have uniformly bounded first derivatives in L^2, the Poincaré inequality gives $\|w - T_\Delta w\|_{L^2} \le (\Delta/\pi)\|w'\|_{L^2}$, corresponding to $r = 1$.

Example 2: polynomial-retaining token. Suppose one token retains coefficients of a low-order polynomial fit on the window. Then $\mathcal{S}_\Delta$ is the corresponding polynomial family, and $T_\Delta w$ is the fitted polynomial surrogate. If the coefficients are chosen by least squares, then $T_\Delta w$ is the best approximation in the induced L^2 norm, so again $C_{\mathrm{qo}} = 1$. For smoother windows, higher-order polynomial surrogates yield faster approximation rates, corresponding to larger values of r.

These examples correspond to the visual cases in Figure 3 and illustrate how T_Δ, $\mathcal{S}_\Delta$, and r are represented in practice.

Proposition A.4 (Token-induced bias component)**.** *Given an oracle map $\phi^\star_\Delta : \mathcal{W}_\Delta \to \mathbb{R}^d$ that is Lipschitz with respect to the window norm $\|\cdot\|_{\mathcal{W}_\Delta}$, i.e., $\|\phi^\star_\Delta(\mathbf{u}) - \phi^\star_\Delta(\mathbf{v})\| \le L_\phi \|\mathbf{u} - \mathbf{v}\|_{\mathcal{W}_\Delta}$, and a token-induced approximation operator T_Δ such that the retained window approximation satisfies $\|\mathbf{w} - T_\Delta \mathbf{w}\|_{\mathcal{W}_\Delta} \le C_{\mathrm{app}} \Delta^r$, the squared norm of the bias term $\|\bar{b}\|^2$ is bounded by*

$$\|\bar{b}\|^2 \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r},$$

where C_{app} measures how well the token-retained representation approximates the original window in $\|\cdot\|_{\mathcal{W}_\Delta}$, and r specifies how that approximation improves as the window size Δ decreases.

Proof. By definition, $b_i = \phi^\star_\Delta(T_\Delta \mathbf{w}_i) - \phi^\star_\Delta(\mathbf{w}_i)$. Given $\|b_i\| \le L_\phi \|T_\Delta \mathbf{w}_i - \mathbf{w}_i\|_{\mathcal{W}_\Delta} \le L_\phi C_{\mathrm{app}} \Delta^r$. Taking the weighted sum and applying the triangle inequality gives

$$\|\bar{b}\| = \left\| \sum_{i=1}^N \omega_i b_i \right\| \le \sum_{i=1}^N \omega_i \|b_i\| \le L_\phi C_{\mathrm{app}} \Delta^r.$$

Squaring both sides yields $\|\bar{b}\|^2 \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r}$. □

Thus, the bias exponent is inherited directly from the approximation rate of the surrogate class.

A.2.2 Bounding the variance

We now analyze the remaining variance term. Throughout this subsection, expectations are taken over the remaining estimation randomness, while the target window statistics are treated as fixed. We first state the rate condition used in the main proof.

Assumption A.5 (Local estimation law)**.** There exist constants $c > 0$ and $p > 0$, independent of the window index and window length over the range considered, such that

$$\mathbb{E}\left[\|\xi_i\|^2\right] \le c\, M_\Delta^{-p}, \qquad M_\Delta = f_s \Delta$$

Assumption A.5 is a statistical rate condition, not a consequence of windowing alone. The exponent p depends on the target window statistic, the noise model, and the estimator used to recover that statistic from M_Δ samples. For example, estimating a window mean gives $p = 1$, while structured parameters such as single-tone frequency can yield faster rates such as $p = 3$.

We give several lemmas below to verify this condition for several common temporal statistics.

Lemma A.6 (Mean estimation)**.** *Consider one window i. Conditional on its underlying mean level μ_i, suppose we observe a temporal sequence as follows*

$$x_i[m] = \mu_i + \varepsilon_i[m], \qquad m = 1, \dots, M_\Delta,$$

where the noise variables are conditionally independent and satisfy

$$\mathbb{E}[\varepsilon_i[m]] = 0, \qquad \mathbb{E}[\varepsilon_i[m]^2] \le \sigma^2$$

Let the target statistic be $\theta_i^\star := \mu_i$, and estimate it by the sample mean, $\hat{\theta}_i := \frac{1}{M_\Delta} \sum_{m=1}^{M_\Delta} x_i[m]$. Then the residual $\xi_i := \hat{\theta}_i - \theta_i^\star$ satisfies $\mathbb{E}\left[|\xi_i|^2\right] \le \sigma^2 M_\Delta^{-1}$ Hence Assumption A.5 holds with $p = 1$.

Proof. Since

$$\xi_i = \hat{\theta}_i - \theta_i^\star = \frac{1}{M_\Delta} \sum_{m=1}^{M_\Delta} \varepsilon_i[m],$$

we have

$$\mathbb{E}[|\xi_i|^2] = \frac{1}{M_\Delta^2} \mathbb{E}\left[\left(\sum_{m=1}^{M_\Delta} \varepsilon_i[m] \right)^2 \right]$$

Expanding the square gives diagonal and cross terms. The cross terms vanish by independence and zero mean, so

$$\mathbb{E}[|\xi_i|^2] = \frac{1}{M_\Delta^2} \sum_{m=1}^{M_\Delta} \mathbb{E}[\varepsilon_i[m]^2] \le \frac{\sigma^2}{M_\Delta}$$

Thus Assumption A.5 holds with $p = 1$. □

Interpretation. In this example, the target window statistic is the mean of a window. Estimating this from M_Δ noisy observations gives the parametric mean-squared error rate M_Δ^{-1}, hence $p = 1$.

Lemma A.7 (Single-tone frequency estimation)**.** *Consider one window i. Assume $\Omega_i \in [\epsilon, \pi - \epsilon]$ for a fixed $0 < \epsilon < \pi/2$. Conditional on its underlying frequency Ω_i, suppose we observe*

$$x_i[n] = A\cos(\Omega_i n + \phi) + \eta_i[n], \qquad n = 0, \dots, M_\Delta - 1,$$

where $A \neq 0$ and ϕ are fixed, $\eta_i[n] \sim \mathcal{N}(0, \sigma^2)$ are independent. Let the target statistic be $\theta_i^\star := \Omega_i$, and let $\hat{\theta}_i := \hat{\Omega}_i$ be an unbiased efficient estimator of Ω_i. For all sufficiently large M_Δ, there exists a constant $C > 0$, depending on A, σ^2, and the frequency region but not on M_Δ, that bounds the mean-squared error as shown below.

$$\mathbb{E}\left[|\xi_i|^2\right] = \mathbb{E}\left[|\hat{\Omega}_i - \Omega_i|^2\right] \le C\, M_\Delta^{-3}$$

Hence Assumption A.5 holds with $p = 3$.

Proof. For Gaussian noise, the Fisher information for Ω_i is

$$\mathcal{I}_{M_\Delta}(\Omega_i) = \frac{1}{\sigma^2} \sum_{n=0}^{M_\Delta - 1} \left(\frac{\partial}{\partial \Omega_i} A\cos(\Omega_i n + \phi) \right)^2 = \frac{A^2}{\sigma^2} \sum_{n=0}^{M_\Delta - 1} n^2 \sin^2(\Omega_i n + \phi).$$

On this compact frequency interval, the oscillatory sum is bounded above and below by constant multiples of M_Δ^3 for sufficiently large M_Δ.

$$\mathcal{I}_{M_\Delta}(\Omega_i) \asymp \sum_{n=0}^{M_\Delta - 1} n^2 \asymp M_\Delta^3$$

For an unbiased estimator,

$$\mathbb{E}\left[|\hat{\Omega}_i - \Omega_i|^2\right] = \mathrm{Var}(\hat{\Omega}_i)$$

The Cramér–Rao bound gives $\mathrm{Var}(\hat{\Omega}_i) \ge c_1 M_\Delta^{-3}$, and efficiency means that the estimator attains this order, i.e., $\mathrm{Var}(\hat{\Omega}_i) \le c_2 M_\Delta^{-3}$ for some constant $c_2 > 0$. Combining them gives the bound. □

Interpretation. Here $p = 3$ is faster than the usual $p = 1$ because the parameter is identified through phase accumulation [85] across the window. The derivative with respect to frequency grows like n, so the Fisher information scales as

$$\mathcal{I}_{M_\Delta}(\Omega_i) \asymp \sum_{n=0}^{M_\Delta - 1} n^2 \asymp M_\Delta^3$$

Since the Cramér–Rao variance bound scales with the inverse Fisher information, an efficient frequency estimator has variance of order M_Δ^{-3}.

Additional examples. Many window statistics used in temporal feature extraction can be categorized into two broad classes. The first class consists of normalized linear summaries, which can satisfy $p = 1$ under the noise and filter conditions in Lemma A.8. The second class consists of structured parameters, such as frequency, for which the signal sensitivity accumulates over time and can have faster rates $p > 1$.

Lemma A.8 (Linear features)**.** *Consider a single window i, and suppose we observe the sequence x_i,*

$$x_i[m] = s_i[m] + \varepsilon_i[m], \qquad m = 1, \dots, M_\Delta,$$

where

$$\mathbb{E}[\varepsilon_i[m]] = 0, \qquad \mathbb{E}[\varepsilon_i[m]\varepsilon_i[\ell]] = 0 \quad (m \neq \ell), \qquad \mathbb{E}[\varepsilon_i[m]^2] \leq \sigma^2$$

Let the target statistic of the window be the linear feature

$$\theta_i^\star := \sum_{m=1}^{M_\Delta} h_{M_\Delta}[m]\, s_i[m], \qquad \hat{\theta}_i := \sum_{m=1}^{M_\Delta} h_{M_\Delta}[m]\, x_i[m],$$

where the weights satisfy

$$\sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 \leq C\, M_\Delta^{-1}$$

Then the residual

$$\xi_i = \hat{\theta}_i - \theta_i^\star = \sum_{m=1}^{M_\Delta} h_{M_\Delta}[m]\varepsilon_i[m]$$

satisfies

$$\mathbb{E}\left[|\xi_i|^2\right] \leq C\sigma^2 M_\Delta^{-1}$$

Hence Assumption A.5 holds with $p = 1$

Proof. By the definition of ξ_i,

$$\xi_i = \sum_{m=1}^{M_\Delta} h_{M_\Delta}[m]\varepsilon_i[m]$$

Therefore,

$$\mathbb{E}\left[|\xi_i|^2\right] = \mathbb{E}\left[\left|\sum_{m=1}^{M_\Delta} h_{M_\Delta}[m]\varepsilon_i[m]\right|^2\right]$$

Expanding the square, the cross terms vanish by conditional uncorrelatedness, so

$$\mathbb{E}\left[|\xi_i|^2\right] = \sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 \mathbb{E}\left[\varepsilon_i[m]^2\right]$$

Using $\mathbb{E}[\varepsilon_i[m]^2] \leq \sigma^2$ and the weight condition yields

$$\mathbb{E}\left[|\xi_i|^2\right] \leq \sigma^2 \sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 \leq C\sigma^2 M_\Delta^{-1}$$

□

Interpretation. Lemma A.8 isolates the common reason some temporal window features satisfy $p = 1$. Their estimation error is a normalized average of M_Δ noise terms. The condition $\sum_m |h_{M_\Delta}[m]|^2 = O(M_\Delta^{-1})$ is the normalization that makes the feature variance decay like M_Δ^{-1}.

Corollary A.9 (Extracting Window mean)**.** *In Lemma A.8, take*

$$h_{M_\Delta}[m] = \frac{1}{M_\Delta}, \qquad m = 1, \dots, M_\Delta$$

Then

$$\theta_i^\star = \frac{1}{M_\Delta} \sum_{m=1}^{M_\Delta} s_i[m], \qquad \hat{\theta}_i = \frac{1}{M_\Delta} \sum_{m=1}^{M_\Delta} x_i[m],$$

and

$$\sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 = M_\Delta \cdot M_\Delta^{-2} = M_\Delta^{-1}$$

Hence

$$\mathbb{E}\left[|\hat{\theta}_i - \theta_i^\star|^2\right] = O(M_\Delta^{-1}), \quad \textit{so} \quad p = 1$$

Corollary A.10 (Fixed spectral coefficients)**.** *In Lemma A.8, any normalized finite-energy filter with*

$$\sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 \le C M_\Delta^{-1}$$

satisfies the same rate $p = 1$. In particular, this covers normalized low-pass averages, smoothed local means, and fixed Fourier coefficients. For the Fourier coefficient at a known bin k,

$$h_{M_\Delta}[m] = \frac{1}{M_\Delta} e^{-2\pi \mathrm{i} k m / M_\Delta},$$

so

$$\sum_{m=1}^{M_\Delta} |h_{M_\Delta}[m]|^2 = M_\Delta \cdot \frac{1}{M_\Delta^2} = M_\Delta^{-1},$$

and therefore

$$\mathbb{E}\left[|\hat{\theta}_{i,k} - \theta_{i,k}^\star|^2\right] = O(M_\Delta^{-1})$$

Interpretation. Corollary A.10 shows that some handcrafted window features already fall under the same $p = 1$. The key point is that the interested frequency or filter shape is known in advance, so the estimator is only projecting the noisy window onto a known direction.

Corollary A.11 (Linear trend on a window)**.** *Given a temporal sequence as follows*

$$x_i[m] = \alpha_i + \beta_i t_m + \varepsilon_i[m], \qquad m = 1, \dots, M_\Delta,$$

Here $t_m = (m-1)/(M_\Delta - 1)$ is an equally spaced normalized time grid with $M_\Delta \ge 2$, and $\varepsilon_i[m]$ satisfy the conditions in Lemma A.8. If the target statistic is the slope $\theta_i^\star := \beta_i$, define the least-squares estimator by

$$(\hat{\alpha}_i, \hat{\beta}_i) \in \arg\min_{\alpha,\beta} \sum_{m=1}^{M_\Delta} \left(x_i[m] - \alpha - \beta t_m\right)^2, \qquad \hat{\theta}_i := \hat{\beta}_i$$

Then

$$\hat{\theta}_i - \theta_i^\star = \hat{\beta}_i - \beta_i = \sum_{m=1}^{M_\Delta} a_m \varepsilon_i[m], \qquad a_m := \frac{t_m - \bar{t}}{\sum_{j=1}^{M_\Delta} (t_j - \bar{t})^2}$$

Consequently,

$$\mathbb{E}\left[|\hat{\theta}_i - \theta_i^\star|^2\right] = \mathbb{E}\left[|\hat{\beta}_i - \beta_i|^2\right] \le \sigma^2 \sum_{m=1}^{M_\Delta} a_m^2 = \frac{\sigma^2}{\sum_{j=1}^{M_\Delta} (t_j - \bar{t})^2}$$

For this grid, $\sum_m (t_m - \bar{t})^2 = M_\Delta(M_\Delta + 1)/(12(M_\Delta - 1))$, so the displayed variance bound is $O(M_\Delta^{-1})$.

$$\mathbb{E}\left[|\hat{\theta}_i - \theta_i^\star|^2\right] = O(M_\Delta^{-1})$$

Thus Assumption A.5 holds with $p = 1$.

Interpretation. The trend estimator also has $p = 1$ because it is still a regular finite-dimensional estimation problem on a normalized time interval. The slope estimation error is a linear combination of the noise terms whose squared coefficients sum to $O(M_\Delta^{-1})$.

Remark A.12 (Slower nonparametric rates). Local estimation rates can also be slower than M_Δ^{-1}. For pointwise estimation of an unknown β-Hölder smooth function from regularly spaced observations with independent Gaussian noise, the standard nonparametric regression model gives the minimax mean-squared rate $M_\Delta^{-2\beta/(2\beta+1)}$ at an interior point [86, 87]. This corresponds to $p = 2\beta/(2\beta+1) < 1$. Other features of the same function can have different estimation rates.

Rates faster than M_Δ^{-1} arise when the parameter influences the signal through an accumulating phase or higher-order structure, so that later samples become increasingly informative. The single-tone frequency in Lemma A.7 is the simplest case as the derivative with respect to frequency grows like n, which leads to Fisher information of order $\sum_{n=0}^{M_\Delta-1} n^2 \asymp M_\Delta^3$ for estimators [88].

More generally, in chirp and polynomial-phase models, higher-order coefficients can exhibit even faster rates. For example, for the chirp model, asymptotic variances of least-squares estimators scale as $O(M_\Delta^{-1})$ for amplitudes, $O(M_\Delta^{-3})$ for frequency, and $O(M_\Delta^{-5})$ for frequency rate [89, 90]. Thus $p > 1$ should be understood as a consequence of increasing temporal sensitivity of the signal.

These lemmas and examples show that Assumption A.5 captures an original property of the different statistical problems rather than an arbitrary definition. The exponent p describes estimation of the specified window statistic under the chosen estimator and noise model.

We now specialize Proposition 3.2 to uniform pooling with uncorrelated residuals. This recovers the original segment-level scaling and separates the local estimation rate from the averaging gain.

Corollary A.13 (Uniform pooling with uncorrelated residuals)**.** *Suppose the centered residuals satisfy Assumption A.5 and $\mathbb{E}\langle \xi_i, \xi_j \rangle = 0$ for $i \neq j$. For uniform weights $\omega_i = 1/N$, with $N = T/\Delta$, the segment-level variance satisfies Equation 10.*

$$\mathbb{E}\|\bar{\xi}\|^2 \leq \frac{a}{T}\Delta^{-(p-1)}, \qquad a := c f_s^{-p}. \tag{10}$$

Proof. Since $M_\Delta = f_s\Delta$, the single-window law gives

$$\mathbb{E}\left[\|\xi_i\|^2\right] \leq c(f_s\Delta)^{-p} = a\Delta^{-p}$$

Using $\bar{\xi} = N^{-1}\sum_{i=1}^N \xi_i$, we expand the squared norm;

$$\mathbb{E}\left[\|\bar{\xi}\|^2\right] = \frac{1}{N^2}\sum_{i=1}^N \mathbb{E}\left[\|\xi_i\|^2\right] + \frac{1}{N^2}\sum_{i\neq j}\mathbb{E}\left[\langle \xi_i, \xi_j\rangle\right]$$

The cross-window terms vanish by conditional uncorrelatedness, so

$$\mathbb{E}\left[\|\bar{\xi}\|^2\right] = \frac{1}{N^2}\sum_{i=1}^N \mathbb{E}\left[\|\xi_i\|^2\right] \leq \frac{1}{N^2}\cdot N \cdot a\Delta^{-p} = \frac{a}{N}\Delta^{-p}$$

Finally, using $N = T/\Delta$, we obtain

$$\mathbb{E}\left[\|\bar{\xi}\|^2\right] \leq \frac{a}{T}\Delta^{-(p-1)}$$

□

Corollary A.13 separates the local estimation rate from the gain obtained by combining windows. Each window contributes a variance bound $a\Delta^{-p}$, while uniform averaging of uncorrelated residuals contributes the factor $1/N = \Delta/T$. Their product gives the segment-level rate $(a/T)\Delta^{-(p-1)}$. For unequal weights, the averaging factor becomes $1/N_{\text{eff}}(\Delta)$. When residuals are correlated, the covariance term in Proposition 3.2 must also be preserved.

A.3 Examples of Error Sources

The decomposition $\hat{\theta}_i - \theta_i^\star = b_i + \xi_i$ is abstract by design. It separates systematic error from stochastic fluctuation without committing to a particular tokenizer, sensor, or downstream task.

To make this split more concrete, Tables 5 and 6 list representative mechanisms that naturally map into the deterministic term b_i and the stochastic term ξ_i, respectively. These examples are illustrative rather than exhaustive, and their purpose is to show how common modeling and measurement effects fit into the two-term decomposition. A concise summary is that b_i aggregates systematic approximation errors induced by tokenization, whereas ξ_i captures centered stochastic perturbations.

Table 5: Representative sources of systematic error b_i.

Source	Mechanism	Effect on b_i
Within-window information loss	Compression of structure	A low-dimensional token might not retain all details of a window, so the same pattern can be systematically under-represented [91].
Boundary effects	Fixed partition cuts events across windows	Events near window boundaries are encoded differently [92, 93] from centered events, creating a deterministic distortion.
Aliasing from coarse aggregation	Resolution too coarse for signal variation	High-frequency (short-timescale) structure is folded into a coarser representation [91] determined by window length Δ.
Temporal misalignment	Event position within the window matters	The same event can be encoded differently depending on its phase inside the window [94].
Within-window nonstationarity	Local trend or drift violates stationarity	A token can approximate a window poorly [95], producing repeatable approximation error.
Scale mismatch	Task depends on sub-window structure	If the downstream target depends on features from finer timescales [96] than Δ, the representation can be intrinsically incomplete.
Architecture bias	Model favors some patterns over others	The tokenizer can smooth or emphasize low-frequency structure and under-represent other task-relevant features [73].

Table 6: Representative sources of stochastic error ξ_i.

Source	Mechanism	Effect on ξ_i
Sensor noise	Thermal or electronic fluctuations	Adds centered perturbations to the measured window and therefore contributes to local random variation [97].
Quantization noise	Discretization of measurements	Introduces zero-mean error when the signal is not locked to quantization levels [98, 99].
Sampling jitter	Random timing perturbations	Small clock fluctuations shift the sampled window slightly and induce estimation error [100].
Measurement noise in time-series sensors	Residual additive or multiplicative fluctuations	Sensor-specific noise that remains after preprocessing appears as stochastic variation around the target statistic [101].
Stochastic optimization noise	Variability around the fitted map	In an abstract learned-representation model, optimization-induced fluctuations can be treated as centered residual variability [102].
Background environmental noise	Ambient perturbations	Fast-changing background effects contribute local random fluctuations after centering [103].
Finite-sample estimation noise	Only M_Δ samples per window	Local feature estimates fluctuate around the target window statistic because they are computed from limited sample support [29, 30].

A.4 Overlapping Windows

We now apply the covariance-aware result to overlapping windows with uniform aggregation weights. Overlap increases the number of window estimates without changing their duration, but shared samples can correlate their errors. Proposition A.14 specializes Proposition 3.2 to this setting and groups the covariance terms by their temporal separation.

Proposition A.14 (Variance under overlapping windows)**.** *Let a sequence of duration $T \geq \Delta$ be covered by windows of duration Δ with stride $\delta \in (0, \Delta]$. For windows fully contained in the sequence, the number of windows is $N_\delta = 1 + \lfloor (T - \Delta)/\delta \rfloor$. Let $\xi_1^{(\delta)}, \ldots, \xi_{N_\delta}^{(\delta)}$ be centered window residuals where we can define $\bar{\xi}_\delta := \frac{1}{N_\delta} \sum_{i=1}^{N_\delta} \xi_i^{(\delta)}$. Then, using the same local estimation law as in Assumption A.5,*

$$\mathbb{E}\left[\|\xi_i^{(\delta)}\|^2\right] \leq a\,\Delta^{-p}, \qquad a := c f_s^{-p}$$

Then

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] = \frac{1}{N_\delta^2} \sum_{i=1}^{N_\delta} \mathbb{E}\left[\|\xi_i^{(\delta)}\|^2\right] + \frac{2}{N_\delta^2} \sum_{1 \leq i < j \leq N_\delta} \mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_j^{(\delta)} \right\rangle\right]$$

Consequently,

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] \leq \frac{a}{N_\delta} \Delta^{-p} + \frac{2}{N_\delta^2} \sum_{1 \leq i < j \leq N_\delta} \mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_j^{(\delta)} \right\rangle\right]$$

If, for each valid lag $h \geq 1$,

$$\mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_{i+h}^{(\delta)} \right\rangle\right] \leq \rho_h^{(\delta)}\, a\, \Delta^{-p},$$

then

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] \leq \frac{a}{N_\delta} \Delta^{-p} \left[1 + 2 \sum_{h=1}^{N_\delta - 1} \left(1 - \frac{h}{N_\delta}\right) \rho_h^{(\delta)}\right]$$

Proof. Using the definition of $\bar{\xi}_\delta$,

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] = \mathbb{E}\left[\left\| \frac{1}{N_\delta} \sum_{i=1}^{N_\delta} \xi_i^{(\delta)} \right\|^2\right]$$

Expanding the squared norm gives

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] = \frac{1}{N_\delta^2} \sum_{i=1}^{N_\delta} \mathbb{E}\left[\|\xi_i^{(\delta)}\|^2\right] + \frac{2}{N_\delta^2} \sum_{1 \leq i < j \leq N_\delta} \mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_j^{(\delta)} \right\rangle\right]$$

Applying the local estimation law to the diagonal terms gives

$$\frac{1}{N_\delta^2} \sum_{i=1}^{N_\delta} \mathbb{E}\left[\|\xi_i^{(\delta)}\|^2\right] \leq \frac{1}{N_\delta^2} N_\delta a \Delta^{-p} = \frac{a}{N_\delta} \Delta^{-p}$$

Keeping the cross-window covariance terms gives the first bound. If the lag-wise covariance bound

$$\mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_{i+h}^{(\delta)} \right\rangle\right] \leq \rho_h^{(\delta)} a \Delta^{-p}$$

holds, then grouping the cross terms by lag yields

$$\frac{2}{N_\delta^2} \sum_{h=1}^{N_\delta - 1} \sum_{i=1}^{N_\delta - h} \mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_{i+h}^{(\delta)} \right\rangle\right] \leq \frac{2a\Delta^{-p}}{N_\delta^2} \sum_{h=1}^{N_\delta - 1} (N_\delta - h) \rho_h^{(\delta)}$$

Factoring out $a\Delta^{-p}/N_\delta$ gives

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] \leq \frac{a}{N_\delta} \Delta^{-p} \left[1 + 2 \sum_{h=1}^{N_\delta - 1} \left(1 - \frac{h}{N_\delta}\right) \rho_h^{(\delta)}\right]$$

This establishes the lag-wise variance bound. □

After completing the general proof for different cases, we give two cases as examples.

Special case: uncorrelated residuals. If the cross-window residuals are uncorrelated, so that

$$\mathbb{E}\left[\left\langle \xi_i^{(\delta)}, \xi_j^{(\delta)} \right\rangle\right] = 0, \qquad i \neq j,$$

then Proposition A.14 reduces to

$$\mathbb{E}\left[\|\bar{\xi}_\delta\|^2\right] \leq \frac{a}{N_\delta}\Delta^{-p}.$$

For long recordings, $N_\delta \approx T/\delta$, so this recovers an approximate δ/Δ reduction relative to nonoverlapping windows. This reduction requires the overlapping windows to contribute uncorrelated residual fluctuations. This case may describe randomness introduced after windowing, such as independent stochastic optimization or independent token-level noise, but it should not be assumed for sample-level noise.

Shared-sample noise example. For physical sensor noise, overlapping windows share noisy samples. To see this, we can consider the scalar mean-estimation with independent sample noise

$$x[m] = s[m] + \varepsilon[m], \qquad \mathbb{E}[\varepsilon[m]] = 0, \qquad \mathbb{E}[\varepsilon[m]^2] = \sigma^2$$

Let $M = f_s\Delta$ be the number of samples per window and $s_\delta = f_s\delta$ be the stride in samples. For the window-mean residual

$$\xi_i^{(\delta)} = \frac{1}{M}\sum_{m=1}^{M} \varepsilon[(i-1)s_\delta + m],$$

we have

$$\mathbb{E}\left[(\xi_i^{(\delta)})^2\right] = \frac{\sigma^2}{M}$$

However, two overlapping windows at lag h share $M - hs_\delta$ samples when $hs_\delta < M$, so

$$\mathbb{E}\left[\xi_i^{(\delta)}\xi_{i+h}^{(\delta)}\right] = \frac{\sigma^2(M - hs_\delta)}{M^2}, \qquad hs_\delta < M$$

Thus the cross-window covariance is positive whenever the windows overlap. For many windows, the factors $1 - h/N_\delta$ are close to one over the lags where windows overlap. When $N_\delta \gg \Delta/\delta$ and Δ/δ is an integer, the leading covariance factor is given by the following identity.

$$1 + 2\sum_{h=1}^{\Delta/\delta - 1}\left(1 - \frac{h\delta}{\Delta}\right) = \frac{\Delta}{\delta}$$

To leading order, this factor cancels the apparent δ/Δ gain from increasing the number of windows. Therefore, for shared sample-level noise, overlapping windows do not create independent new samples and do not necessarily reduce the segment-level variance.

Overlapping windows do not change the window length Δ, so each local estimate is still based on $M_\Delta = f_s\Delta$ samples. The local estimation law and the bias bound therefore remain unchanged. What changes is the number of window estimates being averaged and, crucially, the covariance among those estimates. If the additional overlapping residuals are uncorrelated, overlap reduces the segment-level estimation bound by the factor N/N_δ, which approaches δ/Δ for long recordings. If the residuals arise from shared physical samples, the overlapping estimates can be positively correlated, and the covariance terms can partially or fully cancel this apparent reduction. The variance reduction therefore depends on both the number of estimates and the covariance between their errors.

This distinction is especially relevant for Transformer models, where tokens are not only averaged linearly but are also processed through learned projections, nonlinear transformations, and multi-head attention. Thus, overlap can improve performance by reducing boundary effects and exposing events at multiple relative positions within a window [104]. However, when the residual noise is strongly correlated across overlapping windows, the benefit of these additional views can be weakened, and excessive overlap can even degrade performance due to redundant noisy tokens. This matches our empirical observation that overlapping windows often improve performance, but the gain is not guaranteed when cross-window correlation is high.

B Dataset Details

In this section, we give details about the datasets that are used during our experiments. Overall, we have used 12 datasets with five different time series tasks including *heart rate prediction*, *cardiovascular disease classification*, *step counting*, *activity recognition*, and *sleep stage classification* from four sensor modalities *photoplethysmography*, *electrocardiogram*, and *inertial measurement units*, *electroencephalography*.

For each task, we report the evaluation metrics most commonly used in its corresponding domain. For consistency with the other metrics, we report the correlation coefficient ρ on a 100-point scale, i.e., we multiply ρ by 100.

B.1 Heart Rate Prediction

IEEE SPC. The IEEE SPC dataset overall has 12 subjects, ages ranging from 18 to 58 performing three different activities [105]. Each recording has sampled data from three accelerometer signals and two PPG signals along with the ECG data with a sampling frequency of 125 Hz. All recordings were captured with two 2-channel PPGs with green LEDs, a tri-axial accelerometer, and a chest ECG for the ground-truth. We used all the subjects with leave-one-out cross validation. We averaged the two channels of PPG.

DaLiA. The PPG dataset for motion compensation and heart rate estimation in Daily Life Activities (DaLiA) was recorded from 15 subjects (8 females, 7 males, mean age of 30.6), where each recording was approximately two hours long. PPG signals were recorded while subjects went through different daily life activities, for instance sitting, walking, driving, cycling, working, and so on. PPG signals were recorded at a sampling rate of 64 Hz. We used all the 15 subjects with leave-one-out cross validation.

WildPPG. WildPPG contains 13.5 hours of recordings from 16 participants (total $\approx$216 hours) during outdoor activities, capturing PPG and ECG under real-world conditions [28]. WildPPG dataset includes substantial noise due to motion, temperature fluctuations, and environmental changes, making it especially valuable for assessing algorithm robustness in real-world. We used all the 16 subjects with leave-one-out cross validation.

BIDMC. BIDMC dataset consists of 8-minute resting-state recordings from 53 hospital patients [36], which is a subset of MIMIC-II [106]. PPG waveforms in this dataset are extremely clean compared to real-life recordings, making it well-suited as a proof of concept to demonstrate if methods perform reliably with minimal motion artifacts. We used all the 53 recordings with 10 fold cross validation.

We standardize PPG datasets as follows, same as the previous works [107]. Initially, a fourth-order Butterworth bandpass filter with a range of 0.5–4 Hz is applied. Subsequently, a sliding window of 8 seconds with 2-second shifts is employed for segmentation, followed by z-score normalization of each segments. Lastly, the signal is resampled to a frequency of 25 Hz for each segment.

B.2 Human Activity Recognition

UCIHAR. Human activity recognition using a smartphone's dataset (UCIHAR) [40] is collected with six daily living activities with a waist-mounted smartphone. Six activities include walking, sitting, lying, standing, walking upstairs, and walking downstairs. Data is captured by 3-axial linear acceleration and 3-axial angular velocity at a constant rate of 50 Hz. We used the pre-processing technique the same as in [108] such that the input contains nine channels with 128 features (it is sampled in a sliding window of 2.56 seconds and 50% overlap, resulting in 128 features for each window). Windows are normalized to a mean of zero and unit standard deviation before feeding it to the models. We used the first five subjects as target with leave-one-out-cross validation.

HHAR. Heterogeneity Dataset for Human Activity Recognition (HHAR) is collected by nine subjects within an age range of 25 to 30 performing six daily living activities with eight different smartphones—Although HHAR includes data from smartwatches as well, we use data from smartphones—that were kept in a tight pouch and carried by the users around their waists [41].

Subjects then perform six activities including cycling, sitting, descending stairs, ascending stairs, standing, and walking. Considering the variable sampling frequencies of smart devices in the HHAR dataset, we downsampled the readings to 50 Hz. We employed sliding windows with lengths of 100 (two seconds) and 50, using a specified step size. These windows were then normalized to a mean of zero with unit standard deviation. In our experiments, we utilized the data from the first four subjects (i.e., a, b, c, d) as target domains, following a similar approach to previous papers [27, 109].

USC. USC human activity dataset [42] (USC-HAD) is composed of 14 subjects (7 male, 7 female, aged 21 to 49 with a mean of 30.1) executing 12 activities with a sensor on the front right hip. The data dimension is six (3-axis accelerometer, 3-axis gyroscope). 12 activities include walking forward, walking left, walking right, walking upstairs, walking downstairs, running forward, jumping up, sitting, standing, sleeping, elevator up, and elevator down. We used the pre-processing technique with a smaller window size such that the input contains six channels with 100 features (it is sampled in a sliding window of 1 second and 50% overlap, resulting in 100 features for each window). The last four subjects were used as target domains. The same normalization is also applied to windows before feeding to models.

Clemson. The Clemson dataset has 30 participants, including 15 males and 15 females. Each participant wore three Shimmer3 sensors. We use the data from the device which is positioned on the non-dominant wrist. The inertial measurement data is recorded at 15 Hz. The accelerometers were set to record from -2 to 2 gravities. We computed the overall magnitude of the accelerometer and used it as input to the models. We used the regular gait experiment, where the participant was instructed to walk two laps around a designated path at their normal walking pace. In the semi-regular gait experiment, participants were instructed to perform a scavenger hunt, locating four objects in four different rooms throughout a building [43]. We used 10-fold cross-validation for the evaluation where each fold contains 3 subjects.

B.3 Cardiovascular Disease (CVD) Classification

PhyioNet 2017. The 2017 PhysioNet/CinC Challenge [44] aims to classify, from 8,528 single-lead ECG recordings (between 30 s and 60 s in length), whether the recording shows normal sinus rhythm, atrial fibrillation (AF), an alternative rhythm, or is too noisy to be classified, i.e., four classes. We normalize the signals to have zero mean and unit standard deviation. Additionally, we zero-pad the shorter recordings to ensure they have the same length. We split the dataset into training, validation, and test sets according to the patients using a 60, 20, 20 configuration.

PTB-XL. The PTB-XL dataset is a large public ECG time series dataset recorded from 18,869 subjects, with 12 channels and 5 labels a diagnostic superclass classification problem with labels *NORM*, *MI*, *STTC*, *CD*, and *HYP*. The data split follows the official stratified fold assignment where folds 1–8 are used for training, fold 9 for validation, and fold 10 for testing. We use the 100 Hz version of the sampled ECG signals.

Chapman. Chapman University, Shaoxing People's Hospital (Chapman) ECG dataset which provides 12-lead ECG with a 10-second sampling rate of 500 Hz. The recordings are downsampled to 100 Hz, resulting in each ECG frame consisting of 1000 samples. The labeling setup follows the same approach as in [46] with four classes: atrial fibrillation, GSVT, sinus bradycardia, and sinus rhythm. The ECG frames are normalized to have a mean of 0 and scaled to have a standard deviation of 1. We split the dataset to 80–20% for training and testing as suggested in [46]. We chose leads I, II, III, and V2 during our experiments.

B.4 Sleep Stage Classification

We used the Sleep-EDFx dataset which has five classes including, wake (W), three different non-rapid eye movements (N1, N2, N3), and rapid eye movement (REM). The dataset includes whole-night PSG sleep recordings, where we used a single EEG channel (i.e., Fpz-Cz) with a sampling rate of 100 Hz. We employed non-overlapping 30 s windows and used a 6/2/2 split [49] for training, validation, and testing.

C Baseline Details

C.1 Tokenization

PatchTST. PatchTST is a patch-based Transformer that segments each input channel into fixed-length subseries and treats each patch as a token after a linear layer. Its main tokenization choice is therefore explicit temporal patching, which reduces sequence length and preserves local temporal structure before attention is applied. We include PatchTST as the canonical patch-tokenization baseline [23].

Linear (ov.) patch embeddings. We also consider simple linear patch embeddings built from overlapping windows. These baselines isolate the effect of patch extraction itself by mapping each window directly to a token through a learned linear projection, without adding specialized tokenization or channel independence. They serve as minimal patch-based references alongside PatchTST [23].

Convolutional tokenizers (CvT). We further include convolution-based tokenizers. In CvT, token formation is performed through convolutional token embedding, which replaces a linear patch projection with a convolutional operator prior to Transformer. This method (Ablations in the main section, denoted as *single-kernel*) is used as to evaluate the locality and weight sharing [50].

TOTEM. TOTEM is a tokenizer-centered baseline that learns discrete time-series tokens through self-supervised pretraining. Its tokenizer is based on a tailored VQ-VAE, with a 1D CNN encoder, and decoder trained with a reconstruction objective. The resulting discrete codebook is then used as a tokenizer for downstream evaluation, making TOTEM a natural comparison point for tokenization quality independent of a heavily customized downstream architecture [51].

C.2 Joint Tokenization–architecture Baselines

PathFormer. PathFormer combines tokenization with a dedicated multi-scale Transformer architecture. It divides the input into patches at multiple temporal resolutions, applies dual attention over the resulting patch sequences, and uses adaptive pathways to select and aggregate scale-specific features. We include it as a representative multi-scale patching baseline in which tokenization and architecture are tightly coupled [53].

Medformer. Medformer is a multi-granularity patching Transformer originally proposed for medical time-series classification. Its design combines cross-channel patching, multi-granularity embeddings, and two-stage self-attention across and within granularities. We include it as a baseline where tokenization is co-designed with cross-channel and multi-scale architectural mechanisms [54].

Scaleformer. Scaleformer is an iterative multi-scale refining framework for the Transformer. Rather than relying on a single token scale, it refines predictions across multiple temporal scales with shared weights and scale-specific architectural adaptations. We therefore treat it as a joint tokenization–architecture baseline that incorporates multi-scale structure beyond standard fixed patching [55].

MTST. MTST is a multi-resolution Transformer. It constructs multiple patch-token branches with different patch sizes, processes them through branch-specific attention with positional encoding, and fuses the branch outputs across layers. Because MTST changes both the tokenization and the architecture, we treat it as a joint tokenization–architecture baseline rather than tokenization-only [56].

Kairos. Kairos explicitly addresses the rigidity of fixed patching. Its design combines a dynamic patching tokenizer with instance-adaptive positional embeddings to allow tokenization granularity and temporal encoding to vary with the input. We include Kairos as a recent adaptive-tokenization baseline in which tokenizer design is inseparable from the overall model architecture [57].

Peri-midFormer. Peri-midFormer builds tokens from a periodic pyramid representation rather than from standard temporal patches alone. It decomposes the time series into periodic components at multiple levels and applies attention over inclusion, overlap, and adjacency relations among those components. We include it as a structured tokenization baseline in which token construction is explicitly tied to periodic hierarchy and the downstream attention mechanism [58].

Overall, the first group of baselines isolates token formation itself, whereas the second group couples tokenization with additional architectural changes. This separation is useful for distinguishing gains due to tokenization alone from gains due to broader model redesign.

C.3 Further Discussion of Related Work

Beyond the baselines included in the main experiments, we also conducted additional experiments with other tokenization schemes, including quantization-based tokenization like Chronos [75]. In these approaches, the input is tokenized at the level of individual timestamps (i.e., window length 1), and the values are discretized before being passed to the Transformer. Compared with duration-based tokenization, this strategy places a stronger burden on the model, since adjacent scalar values are first mapped to discrete token embeddings and the geometry of the continuous signal must be recovered during training [110]. In noisy settings, this discretization can be particularly unstable. In our experiments, these methods were not competitive with the strongest baselines, particularly on the noisiest datasets, where the discretized representations were less stable. Since these methods were not competitive and were not designed primarily as pure tokenization baselines, we did not include them in the main comparison.

WaveToken [111] is a recent tokenizer which couples the method with a full forecasting pipeline, including z-score scaling, discrete wavelet decomposition, thresholding, quantization, and an autoregressive encoder-decoder. This makes it difficult to isolate the contribution of tokenization alone from the contribution of the downstream architecture and training setup. While wavelets provide sparse multi-resolution representations [112, 113], the resulting coefficients are produced by filtering and downsampling, so their temporal support is distributed rather than aligned with disjoint temporal windows. As a result, the attention mechanism must still recover the timing of events (if they are related to the downstream task) from overlapping coefficients, which makes the tokenization less interpretable in time and more challenging for optimization. We attempted to reproduce the method using the repository linked in the paper. The paper points to the Chronos forecasting repository, but a later GitHub discussion indicates that the code release had been deprioritized. Because we could not verify a stable official implementation and our reimplementation was not competitive, we did not include WaveToken in the final comparison.

We also experimented with additional baselines, including iTransformer [78] and Crossformer [114]. These methods were mainly developed for multivariate forecasting, whereas several of our datasets, such as DaLiA, WildPPG, and PhysioNet, are single-channel. As expected, their performance was generally weak in those settings. We also found them to be non-competitive on most multi-channel datasets, with the main exception of Crossformer on PTB-XL. Since these baselines were not competitive and pure tokenization comparison, we did not include them in the main comparison.

There is also a growing line of work on tokenizing or discretizing continuous temporal sequences for use with large language models (LLMs) [115]. Methods such as Chronos and ChatTime [116] convert real-valued time series into discrete token sequences through normalization and quantization, and then train or adapt Transformer-based language models on the resulting vocabulary. Related work on numerical tokenization in LLMs has emphasized that the choice of representation for continuous values can strongly affect learning and generalization [79]. Although our method could be extended in this direction and combined with LLM-style pretraining, our focus in this paper is on tokenization for representation learning rather than alignment with a pre-existing language-model vocabulary. We therefore leave the study of LLM-oriented tokenization as future work.

D Implementation Details

We conducted experiments on NVIDIA GeForce RTX 4090 GPUs, involving training with three random seeds for all datasets, totaling approximately 800 GPU hours including ablations and synthetic experiments. This section summarizes the implementation details for the proposed method and the baselines. We begin with the proposed tokenizer and then give details about the baselines.

D.1 Algorithm

In this section, we present the pseudocode implementation for the proposed tokenizer. Given an input sample $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times T f_s}$, the tokenizer outputs

$$\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_N] \in \mathbb{R}^{N \times d},$$

where $P = M_\Delta = f_s \Delta$ is the window (patch) length in samples, S is the stride, and N is the number of tokens. To avoid evaluating the K views separately, we write the tokenizer as a vectorized masked convolution bank. The tokenizer uses K views with odd kernel lengths $\{k_m\}_{m=1}^{K}$. We use unit dilation. Each view is embedded into a common support length

$$R = \max_{m=1,\ldots,K} k_m,$$

and a binary mask restricts the trainable weights of view m to the centered k_m locations within the length-R support.

Algorithm 1 Vectorized multitaper tokenization.

Input: $\mathbf{x} \in \mathbb{R}^{B \times C_{\text{in}} \times T f_s}$, patch length P, stride S, masked bank weights $\mathbf{W}_{\text{bank}} \in \mathbb{R}^{(Kd) \times C_{\text{in}} \times R}$, binary mask $\mathbf{M}_{\text{bank}} \in \{0,1\}^{(Kd) \times C_{\text{in}} \times R}$, gate parameters $\mathbf{g} \in \mathbb{R}^K$

Output: $\mathbf{Z} \in \mathbb{R}^{B \times N \times d}$

$L \leftarrow T f_s, \quad q \leftarrow (S - (L \bmod S)) \bmod S$

if $q > 0$ **then**

    Right-pad $\mathbf{x}$ by q samples

end if

$N \leftarrow (L + q)/S$

$\mathbf{W}_{\text{eff}} \leftarrow \mathbf{W}_{\text{bank}} \odot \mathbf{M}_{\text{bank}}$ ▷ zeroes weights outside each support

$\mathbf{Y} \leftarrow \text{Conv1D}(\mathbf{x}; \mathbf{W}_{\text{eff}}, \text{stride} = S, \text{padding} = \lfloor R/2 \rfloor)$ ▷ $\mathbf{Y} \in \mathbb{R}^{B \times (Kd) \times N}$

$\mathbf{Y} \leftarrow \text{reshape}(\mathbf{Y}, B, K, d, N)$

$\boldsymbol{\alpha} \leftarrow \sigma(\mathbf{g})$ ▷ learned gates

$\widetilde{\mathbf{Z}}_{b,n,\ell} \leftarrow \sum_{m=1}^{K} \alpha_m \, \mathbf{Y}_{b,m,\ell,n}$ ▷ vectorized contraction

$\mathbf{Z} \leftarrow \text{LayerNorm}(\widetilde{\mathbf{Z}})$

Return: $\mathbf{Z}$

Algorithm 1 summarizes the proposed tokenization. We use scalar gates so that each view contributes one global weight, keeping the fusion mechanism lightweight. Because all views are evaluated on the same token grid, summation occurs across views at the same token location n, so the number of tokens passed to the Transformer remains unchanged. We use weighted summation rather than concatenation to preserve the token dimension d; concatenation would increase the feature dimension by a factor of K, thereby changing the backbone and confounding comparisons with the base architecture.

We use odd kernel lengths so that symmetric padding preserves token-grid alignment. Thus, the implemented weight bank $\mathbf{W}_{\text{bank}}$ remains compatible with symmetric padding and a shared token grid while being efficient.

D.2 Architecture

Here, we present the details of the architecture that is used for the experiments. After tokenization, the resulting sequence is processed by a standard encoder-only Transformer. Given an input sequence $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times T f_s}$, the tokenizer produces a sequence of N tokens in $\mathbb{R}^d$. We then prepend a learned classification token to this sequence and add positional encodings before applying the encoder.

Our Transformer uses a pre-normalization design with residual connections. Each layer consists of a multi-head self-attention block [1] followed by a feedforward block. The feedforward block is implemented as a two-layer MLP with ReLU activation and dropout [117]. After the final layer, we use the output corresponding to the classification token as the sequence-level representation for downstream task. In our main implementation, the Transformer backbone uses embedding dimension $d = 64$, depth $L = 6$, 4 attention heads, feedforward hidden dimension 128, and dropout rate 0.1. The tokenizer output dimension matches the Transformer embedding dimension, so no additional projection is required between tokenization and the encoder. We further examine robustness of results to these design choices in Appendix E, especially to the token dimension d, since it determines how much information from each window is compressed into a single token.

D.3 Baselines

Here, we give details about the implementation of baselines. We focus on the deep learning methods as their implementation requires setting hyperparameters. Also, since some of the works are mainly proposed for forecasting, we have explained how to use their representations for downstream tasks.

TOTEM We use the official released code in (`https://github.com/SaberaTalukder/TOTEM`). TOTEM learns a discrete tokenizer through a self-supervised VQ-VAE, consisting of a 1D convolutional encoder and convolutional decoder trained with a reconstruction objective. The learned tokenizer is then frozen and used to produce discrete latent representations for downstream tasks.

For each split, the tokenizer was trained only on the training data to avoid leakage into validation or test sets. We used the TOTEM VQ-VAE backbone with details (i.e, hidden and latent dimension) in Table 7. The residual stack used 4 residual layers with residual hidden size 64, the commitment cost was set to 0.25, the batch size was 64, and the learning rate was 3×10^{-4}. After training, we used the learned tokenizer as a fixed module. In our study, the same downstream Transformer backbone used with TOTEM tokens for a fair comparison.

Pathformer We implemented PathFormer using the official codebase (`https://github.com/decisionintelligence/pathformer`) and retained its multi-scale architecture, including multi-scale patch division, dual attention over patches, and adaptive pathway routing [53]. Because the released model is designed for forecasting, we adapted it to sequence classification by replacing the original prediction head with a classification head.

We apply the input projection and pass the representation through the stacked AMS blocks. The expert patch size, $\mathrm{Exp_P}$, and other hyperparameters are listed in Table 7. After the final AMS block, we average-pool, flatten across channels, and apply an MLP classifier composed of layer normalization, a linear layer to dimension d, GELU, dropout with rate 0.3, and a final linear layer to the number of classes. Reversible Instance Normalization (RevIN) [118] is implemented before the AMS stack. We evaluated both variants, with and without RevIN, and observed small but consistent gains when RevIN was enabled. The PathFormer implementation returns an auxiliary loss from each AMS block. In our experiments, we follow the implementation, and sum these terms across layers and add to the training loss.

MedFormer We implemented Medformer using the official codebase (`https://github.com/DL4mHealth/Medformer`). In our experiments, we retained the official Medformer backbone and used its native classification rather than replacing the model with a classifier head as its main objective is also classification. We use the patch lengths $\mathrm{P_length}$ listed in Table 7. Since Medformer uses separate branches for different patch scales prior to cross-scale interaction, its total parameter count is higher than that of a standard single-scale Transformer with comparable backbone settings. This additional cost arises from the scale-specific projection and attention modules required by the multi-granularity architecture.

ScaleFormer. We implemented ScaleFormer using the official codebase (`https://github.com/BorealisAI/scaleformer`). We retained the multi-scale refine-

ment blocks and used the scale configurations listed in Table 7. For classification tasks, the final temporal representation was pooled and passed to an MLP classifier.

MTST We implemented MTST using the public codebase (`https://github.com/networkslab/MTST`), which provides the official implementation for Multi-resolution Time-Series Transformer [56]. MTST builds multi-resolution temporal representations using multiple patch-tokenization branches and Transformer components, rather than only replacing the first embedding layer. We retained its multi-resolution Transformer components and adapted the output head for our downstream tasks. Specifically, we evaluated both a learnable `[CLS]` token and an MLP head applied to the extracted temporal features, where the MLP outputs class logits. Since the `[CLS]` gave better performance, we report the results accordingly.

Kairos We implemented Kairos using the official codebase from GitHub (`https://github.com/foundation-model-research/Kairos`). We integrated the Kairos backbone into our classification pipeline through a module. Since Kairos consumes one-dimensional input series, we encode each channel independently and then aggregate the resulting channel-wise representations (if there is more than one channel) into a single embedding. More precisely, for an input $\mathbf{x} \in \mathbb{R}^{C_{\text{in}} \times T f_s}$, we reshape it into C_{in}, pass them through the encoder, mean-pool the output tokens to obtain one feature vector per channel. Then, the resulting representations are fed to an MLP classifier. The hyperparameters, i.e., number of activated experts, are listed in Table 7.

Peri-midFormer We implemented Peri-midFormer using the official codebase (`https://github.com/WuQiangXDU/Peri-midFormer`) and retained its core periodic-pyramid design. Since the released model supports classification, we used its classification pathway.

The hyperparameters, including the number of selected periods $\mathrm{top_k}$, the hidden dimension d, the moving-average are listed in Table 7. Its parameter count is higher than that of a standard Transformer with comparable depth and hidden dimension, since the periodic decomposition and multi-component interaction modules introduce additional scale and relation-specific computation.

To ensure a fair comparison, we performed hyperparameter studies for baselines. Because deep learning baselines introduce many tunable parameters, we followed a structured search scheme. We first evaluated a plain Transformer with linear patch mapping, following PatchTST, using patch sizes chosen as powers of two $\{1, 2, 4, 8, 16, 32, 64, 128, 256\}$. We then used the best-performing patch size to define method-specific grid searches around that scale. For instance, for Medformer, we considered scale sets such as $\{16, 32, 64\}$ or $\{32, 64, 128\}$, selected according to the best patch size and its half scales with neighbors. We did not consider repetitive scale configurations such as $\{2, 2, 4, 4, 8, 8, 16, 32\}$, since they both increase the computational cost of the backbone and make the search space prohibitively large. We applied the same procedure to the other baselines, including the expert lengths in PathFormer and the main window size in Kairos.

For PathFormer, although RevIN was designed for forecasting, we found that it also provided slight gains for classification and discrimination, so we kept it in our experiments. For the auxiliary loss weight, we searched over 10^{-2} and 10^{-3}. We observed that PathFormer was the heaviest baseline in terms of both model size and convergence time.

For Kairos, we observed that the best performance is achieved when the main window size is set to roughly twice, or more, than the optimal window size found in the plain Transformer setting. An explanation is the hierarchical mechanism of Kairos where the model processes a coarser window and then routes it to experts for finer-grained analysis. If the main window is already small, the experts operate on sub-windows that are themselves suboptimal, which limit the benefit of the multi-experts.

For the experiments, we used a hidden dimension of 64. We also performed experiments by increasing it but we have not observed improvements. However, for ECG and EEG, larger hidden dimensions benefited some baselines. In those cases, we doubled the hidden dimension for the corresponding baselines, which also led to a substantial increase in the number of parameters.

Overall, the large number of hyperparameters in these baselines is a drawback. In our experiments, tuning them required substantial time and computational resources, and careful search was necessary.

For PatchTST and its overlapping variant, we optimized the hyperparameters for each dataset, since these models are relatively cheap to tune. This gives them a stronger setting than using a single task-level choice shared across datasets. For multichannel inputs in PatchTST, we evaluated both channel-independent and joint channel-to-token mapping, and used the better strategy for

Table 7: Main hyperparameters for the methods on each dataset

(a) Heart rate estimation using PPG

Method	*IEEE SPC*	*DaLiA & WildPPG*	*BIDMC*
Tokenization baselines			
Linear	$M_\Delta = 32$	$M_\Delta = 4$ & $M_\Delta = 8$	$M_\Delta = 64$
Linear (ov.)	$M_\Delta = 16$, $S = 4$	$M_\Delta = 8$, $S = 4$	$M_\Delta = 8$, $S = 4$
TOTEM	epochs = 50, hidden = 128, latent = 64, codebook = 512		
Multitaper	$k_m = \{9,\ 17,\ 33\}$, $S = 4$		
Joint tokenization–architecture baselines			
PathFormer	Exp_P $= \{8, 4, 2\}$, $k = 4$, $d = 64$, $L = 2$, $d_{\text{ff}} = 128$		
Medformer	P_length $= \{4, 16, 32\}$ $d = 64$, $L = 4$, $H = 4$ $d_{\text{ff}} = 128$, dropout $= 0.1$		
Kairos	$n_{\text{act}} = 3$ patch $= 16$, levels $= 3$ $d = 64$, $L = 8$, $H = 4$		
Peri-midFormer	top_k $= 3$, moving_avg $= 25$ $d = 64$, $L = 2$, $H = 4$	top_k $= 3$, moving_avg $= 15$ $d = 64$, $L = 2$, $H = 4$	top_k $= 3$, moving_avg $= 25$ $d = 64$, $L = 2$, $H = 4$
MTST	P_length $= \{4, 8, 16\}$, strides $= \{2, 4, 8\}$, $d = 64$, $L = 4$, $H = 4$, $d_{\text{ff}} = 128$, dropout $= 0.1$		
ScaleFormer	scales $= \{8, 4, 2, 1\}$, $d = 64$, $L = 8$, $H = 4$, $d_{\text{ff}} = 128$, dropout $= 0.1$		

(b) Human activity recognition and Step counting using IMUs

Method	*UCIHAR*	*HHAR & USC*	*Clemson*
Tokenization baselines			
Linear	$M_\Delta = 16$	$M_\Delta = 2$ & $M_\Delta = 16$	$M_\Delta = 8$
Linear (ov.)	$M_\Delta = 8$, $S = 4$	$M_\Delta = 8$, $S = 4$	$M_\Delta = 64$, $S = 32$
TOTEM	epochs = 50, hidden = 128, latent = 64, codebook = 512		
Multitaper	$k_m = \{5,\ 9\}$, $S = 2$	$k_m = \{5,\ 9\}$, $S = 2$	$k_m = \{5,\ 9\}$, $S = 2$
Joint tokenization–architecture baselines			
PathFormer	Exp_P $= \{8, 4, 2\}$, $k = 4$, $d = 64$, $L = 2$, $d_{\text{ff}} = 128$		
Medformer	P_length $= \{4, 16, 32\}$, $d = 64$, $L = 4$, $H = 4$, $d_{\text{ff}} = 128$, dropout $= 0.1$		
Kairos	$n_{\text{act}} = 3$, patch $= 16$, levels $= 3$, $d = 64$, $L = 8$, $H = 4$		
Peri-midFormer	top_k $= 3$, moving_avg $= 25$ $d = 64$, $L = 2$, $H = 4$	top_k $= 3$, moving_avg $= 15$ $d = 64$, $L = 2$, $H = 4$	top_k $= 3$, moving_avg $= 25$ $d = 64$, $L = 2$, $H = 4$
MTST	P_length $= \{2, 4, 8\}$, strides $= \{1, 2, 4\}$, $d = 64$, $L = 4$, $H = 4$, $d_{\text{ff}} = 128$		P_length $= \{8, 16, 32\}$
ScaleFormer	scales $= \{8, 4, 2, 1\}$, $d = 64$, $L = 8$, $H = 4$, $d_{\text{ff}} = 128$, dropout $= 0.1$		

(c) Cardiovascular disease classification using ECG and sleep stage classification using EEG

Method	*PhysioNet*	*PTB-XL & Chapman*	*Sleep*
Tokenization baselines			
Linear	$M_\Delta = 128$	$M_\Delta = 4$ & $M_\Delta = 32$	$M_\Delta = 64$
Linear (ov.)	$M_\Delta = 128$, $S = 16$	$M_\Delta = 8$, $S = 4$	$M_\Delta = 32$, $S = 8$
TOTEM	epochs = 100, hidden = 128, latent = 64, codebook = 1024		
Multitaper	$k_m = \{33, 65, 129\}$, $S = 8$	$k_m = \{33, 65, 129\}$, $S = 8$	$k_m = \{33, 65, 129\}$, $S = 8$
Joint tokenization–architecture baselines			
PathFormer	Exp_P $= \{50, 20, 10\}$, $k = 4$, $d = 64$, $L = 2$, $d_{\text{ff}} = 128$		
Medformer	P_length $= \{16, 64, 128\}$, $d = 64$, $L = 4$, $H = 4$, $d_{\text{ff}} = 128$, dropout $= 0.1$		P_length $= \{16, 64, 128\}$
Kairos	$n_{\text{act}} = 3$, patch $= 32$, levels $= 3$, $d = 64$, $L = 8$, $H = 4$		
Peri-midFormer	top_k $= 4$, moving_avg $= 15$ $d = 128$, $L = 2$, $H = 4$	top_k $= 3$, moving_avg $= 15$ $d = 64$, $L = 2$, $H = 4$	top_k $= 4$, moving_avg $= 15$ $d = 128$, $L = 2$, $H = 4$
MTST	P_length $= \{8, 16, 32\}$, strides $= \{2, 4, 8\}$, $d = 64$, $L = 3$, $H = 2$, $d_{\text{ff}} = 128$		P_length $= \{16, 32, 64\}$
ScaleFormer	scales $= \{16, 8, 4, 2\}$, $d = 128$, $L = 8$, $H = 4$, $d_{\text{ff}} = 256$, dropout $= 0.1$		

multichannel datasets. When the window size is fixed at the task level rather than selected per dataset, performance drops, further highlighting the sensitivity of tokenization. In Table 7, we report only the hyperparameters that vary across tasks in each section for each model, to save space. For example, if the number of layers and heads remain the same across tasks and only the window length changes, then only the length is listed in that column.

E Robustness and Significance Analysis of Our Method

We also performed additional experiments to assess the robustness of our method to hyperparameter choices. In particular, we vary the token embedding dimension and the multitaper filter lengths k_m.

During experiments, our method uses an embedding dimension of 64. Here, we evaluate both a halved and a doubled version of this dimension to examine how performance changes with the size of the token space.

For the multitaper kernels k_m, we ablate the main configuration (Table 7) by modifying the lengths within the same set, $\{5, 9, 17, 33, 65, 129\}$. In k_m (down), we replace the longest window with the next shorter candidate, whereas in k_m (up), we replace the shortest window with the next longer.

We also evaluated nearby stride settings by decreasing ($S \downarrow$) and increasing ($S \uparrow$) the selected stride by one step in the candidate set. For example, when the selected stride was $S = 8$, we additionally tested $S = 4$ and $S = 16$.

In our original implementation, the tokenizer mapping function is parameterized using convolutional kernels. Although this mapping is, in principle, related to a linear projection as a mathematical operation, we also replaced it with an explicit linear mapping (Linear $\hat{\phi}(\mathbf{w})$) to examine whether the choice of mapping function affects performance. The resulting comparisons are reported together with the other baselines for completeness.

Table 8: Performance comparison of tokenization methods for heart rate estimation using PPG.

Method	IEEE SPC			WildPPG			DaLiA			BIDMC		
	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑
Linear	10.11±0.31	17.77±0.54	68.89±2.25	9.92±0.04	17.82±0.03	41.50±0.27	5.67±0.11	12.03±0.18	78.06±0.52	1.76±0.02	4.51±0.07	89.67±0.06
Linear (ov.)	8.10±0.11	16.74±0.35	73.57±1.07	9.96±0.04	17.91±0.44	41.19±0.37	5.90±0.10	11.94±0.29	78.30±0.87	1.67±0.05	4.11±0.40	89.62±2.23
PathFormer	18.52±1.08	24.66±1.50	26.70±9.72	13.95±0.37	19.18±0.33	14.16±2.66	—	—	—	2.70±0.15	5.30±0.20	89.48±0.65
MedFormer	7.80±0.34	15.60±0.59	72.86±2.16	10.08±0.01	17.84±0.02	39.98±0.38	5.83±0.10	12.62±0.19	75.88±0.41	2.21±0.16	5.22±0.53	85.24±3.50
Totem	18.79±0.44	24.54±0.71	37.07±8.28	—	—	—	—	—	—	4.58±0.13	8.04±0.12	72.43±1.31
PerimidFormer	23.06±1.03	30.96±1.31	14.27±2.04	14.61±0.38	22.66±0.54	19.82±2.44	17.67±0.23	24.94±0.46	28.63±1.36	9.58±0.48	9.95±0.50	20.23±4.52
ScaleFormer	12.88±0.06	17.15±0.19	69.79±0.73	10.73±0.16	18.13±0.16	35.29±1.76	6.65±0.71	13.09±0.21	74.26±0.65	3.04±0.07	5.26±0.07	89.51±0.18
MTST	17.35±0.34	22.68±0.59	60.28±1.26	13.29±0.10	18.43±0.32	18.96±2.27	9.90±0.64	16.45±1.14	68.12±1.35	3.95±0.29	9.14±0.73	65.43±0.06
Kairos	8.26±0.32	16.30±0.53	73.89±1.25	10.21±0.09	18.00±0.13	39.39±0.67	6.08±0.05	12.58±0.08	75.29±0.03	1.96±0.20	4.87±0.58	88.03±3.88
Multitaper	6.50±0.32	**12.80**±0.54	**82.32**±1.13	**9.75**±0.06	**17.63**±0.05	42.15±0.19	**5.42**±0.10	**11.50**±0.10	78.58±0.13	1.58±0.08	4.04±0.22	91.10±1.03
– Dim (32)	6.33±0.11	13.35±0.29	81.00±1.62	9.98±0.07	17.81±0.11	41.40±0.47	**5.42**±0.10	11.57±0.20	78.50±0.54	1.64±0.13	4.47±0.62	87.67±3.72
– Dim (128)	6.67±0.06	13.44±0.27	79.94±0.48	10.29±0.06	18.21±0.05	40.33±0.37	5.91±0.05	12.27±0.13	76.36±0.11	**1.55**±0.07	**3.81**±0.22	91.99±0.79
– k_m (down)	7.96±0.21	13.04±0.24	79.04±0.39	9.97±1.22	17.99±0.36	41.38±0.09	5.83±0.07	11.68±0.13	78.04±0.44	1.78±0.02	4.02±0.24	**93.55**±0.92
– k_m (up)	**6.26**±0.25	12.90±0.95	81.75±1.95	9.88±0.09	17.75±0.12	41.92±0.55	5.63±0.04	11.93±0.05	77.57±0.24	1.61±0.06	4.33±0.31	89.45±1.50
– $S \downarrow$	6.50±0.13	13.10±0.30	81.97±0.82	9.84±0.12	17.71±0.23	41.86±0.81	6.18±0.05	12.40±0.14	76.16±0.52	1.75±0.08	4.64±0.44	90.62±1.10
– $S \uparrow$	6.92±0.13	13.50±0.46	81.03±0.82	9.90±0.03	17.79±0.04	41.20±0.37	5.59±0.06	11.66±0.15	77.98±0.26	1.60±0.07	4.19±0.03	90.51±1.67
– Linear $\hat{\phi}(\mathbf{w})$	7.10±0.31	14.36±0.52	79.46±0.22	9.83±0.04	17.75±0.08	**42.31**±0.33	5.55±0.10	**11.50**±0.10	**79.20**±0.22	1.62±0.45	4.37±0.16	90.78±1.02

Table 9: Performance comparison of tokenization methods for activity and step using IMU.

Method	UCIHAR			HHAR			USC			Clemson		
	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	MAE ↓	RMSE ↓	ρ ↑
Linear	87.62±0.68	87.39±0.50	86.35±0.78	81.06±0.67	80.13±0.57	80.19±0.53	67.40±5.27	63.66±6.12	65.93±6.02	3.22±0.15	4.41±0.18	11.74±0.11
Linear (ov.)	87.95±0.75	87.87±0.85	87.01±1.02	82.62±0.95	81.77±0.84	82.36±1.07	71.83±0.80	69.13±1.23	70.47±1.40	1.96±0.13	3.24±0.18	59.28±4.56
PathFormer	70.12±0.35	64.89±3.66	66.16±4.51	50.85±4.72	46.23±5.67	47.32±5.79	51.47±2.26	48.14±2.21	48.77±2.55	2.39±0.26	3.59±0.28	44.92±6.55
MedFormer	83.98±5.50	83.24±5.57	82.58±6.98	77.26±1.15	76.11±1.42	75.96±1.58	67.89±3.09	68.45±3.22	66.19±3.60	1.91±0.21	3.26±0.16	57.57±6.09
Totem	79.87±0.29	80.12±0.07	80.07±0.16	—	—	—	62.54±3.06	68.42±0.40	67.03±1.26	—	—	—
PerimidFormer	80.00±8.13	78.96±9.12	79.13±9.04	68.07±3.53	65.93±3.79	66.82±3.64	59.90±6.67	56.60±6.38	57.88±7.00	3.41±0.13	4.46±0.18	5.67±0.31
ScaleFormer	88.54±2.63	87.95±3.14	87.52±3.33	81.66±0.98	80.44±0.88	81.22±0.93	71.90±2.30	65.67±1.32	68.24±1.96	3.46±0.20	4.78±0.27	7.16±5.40
MTST	73.14±1.47	71.07±1.62	70.22±1.59	72.67±1.04	70.89±0.83	71.36±1.14	59.56±2.34	54.28±2.60	56.26±3.26	3.80±0.67	4.52±0.73	10.36±1.97
Kairos	69.38±0.97	67.15±0.49	68.28±0.04	70.20±1.15	68.58±1.03	69.67±0.86	54.86±2.02	51.09±3.02	52.16±2.46	1.61±0.08	3.03±0.15	60.40±4.97
Multitaper	91.37±2.49	91.30±2.23	90.89±2.68	80.12±1.50	79.23±0.69	80.87±0.68	74.69±1.78	71.60±1.99	73.25±2.12	1.68±0.46	2.92±0.35	61.53±1.90
– Dim (32)	90.07±0.79	89.97±0.85	88.93±1.20	77.36±1.09	77.12±1.39	76.83±1.50	73.38±2.20	69.81±2.13	72.12±2.32	1.88±0.85	3.06±0.76	57.51±0.27
– Dim (128)	92.40±1.90	**92.36**±1.89	**92.19**±2.53	77.73±0.82	76.81±0.92	77.13±0.97	73.29±1.02	70.69±1.40	72.42±1.10	1.67±0.29	2.89±0.29	63.60±9.90
– k_m (down)	91.52±2.71	90.98±3.16	90.77±3.89	77.35±1.66	76.28±1.89	76.56±1.97	73.58±1.32	71.26±1.50	72.50±1.54	2.28±0.70	3.44±0.68	46.40±3.08
– k_m (up)	91.25±3.08	91.14±2.85	90.56±3.62	76.65±0.75	75.64±0.83	75.92±0.91	73.22±2.91	70.10±2.97	71.99±2.85	**1.38**±0.26	**2.65**±0.19	**69.27**±8.23
– $S \downarrow$	91.29±2.90	90.69±2.90	90.49±3.42	77.20±0.32	76.19±0.51	76.50±0.45	74.46±1.71	70.77±2.59	73.39±3.25	1.78±0.04	2.96±0.33	58.96±1.98
– $S \uparrow$	90.74±2.98	90.12±2.88	90.12±3.20	77.10±0.70	76.17±0.44	76.00±0.80	73.10±3.18	70.64±3.10	72.08±3.70	2.20±0.46	3.34±0.39	50.55±1.84
– Linear $\hat{\phi}(\mathbf{w})$	**92.52**±1.65	92.14±1.89	92.11±1.90	**83.31**±2.34	**82.26**±2.95	**83.05**±2.58	**75.51**±1.00	**72.83**±0.78	**74.42**±0.94	1.65±0.27	2.96±0.33	66.11±7.85

As shown by these results, our method is robust to changes in the hyperparameters. In some datasets, modifying the filter lengths or mapping function can even lead to performance improvements. However, during our experiments, we fixed the same set of lengths within each task rather than tuning them separately for each dataset, in order to better assess the generalization of the method.

Table 10: Performance comparison of tokenization methods for CVD and sleep using ECG and EEG.

Method	PhysioNet 2017			PTB-XL			Chapman			Sleep		
	Acc ↑	F1 ↑	W-F1 ↑	AUROC ↑	F1 ↑	Acc ↑	Acc ↑	F1 ↑	W-F1 ↑	κ ↑	F1 ↑	W-F1 ↑
Linear	68.03±0.29	50.54±1.40	65.92±0.66	91.10±0.31	70.20±0.55	**60.94**±1.15	96.88±0.20	89.28±1.11	96.83±0.19	71.75±0.51	63.75±0.22	85.61±0.14
Linear (ov.)	70.50±1.11	56.86±2.97	68.29±1.70	91.39±0.22	71.18±0.43	60.91±0.73	97.55±0.14	91.10±0.17	97.54±0.14	73.04±0.71	65.03±0.50	84.65±0.32
PathFormer	59.20±0.03	18.79±0.27	44.25±0.31	—	—	—	70.12±2.57	30.21±5.47	60.07±7.51	—	—	—
MedFormer	68.35±1.01	55.11±1.83	66.96±0.89	90.71±0.08	70.27±0.19	60.27±0.71	95.40±0.38	85.53±0.23	94.48±0.41	72.13±0.52	64.20±0.30	86.03±0.17
Totem	61.94±1.01	41.65±2.23	56.77±0.69	—	—	—	96.93±0.13	77.72±0.94	96.50±0.14	—	—	—
PerimidFormer	60.12±0.03	18.84±0.35	44.30±0.39	82.54±0.58	55.74±2.37	44.38±2.47	84.75±3.10	49.38±6.75	84.63±3.90	62.19±1.25	52.84±2.22	82.02±0.70
ScaleFormer	59.16±0.06	18.59±0.01	44.01±0.03	90.59±0.37	69.27±1.33	60.52±0.71	94.65±0.80	77.16±2.27	94.25±0.97	67.64±0.14	56.07±0.33	84.25±0.12
MTST	62.11±1.81	29.36±1.53	48.24±0.06	83.41±0.49	58.71±1.30	56.01±1.62	96.41±0.26	71.64±1.13	95.54±0.33	66.17±0.20	55.28±0.96	82.04±0.59
Kairos	68.93±1.72	56.26±0.11	66.46±1.39	88.25±0.45	66.38±1.03	57.04±0.86	97.94±0.08	91.36±0.35	97.98±0.08	73.06±0.65	63.55±0.76	86.22±0.39
Multitaper	73.82±0.96	64.04±1.60	72.84±0.68	90.76±0.03	69.14±0.37	60.04±0.97	98.37±0.39	94.55±1.08	98.37±0.39	75.40±0.30	66.82±0.34	87.52±0.09
– Dim (32)	73.82±0.64	63.62±0.61	72.84±0.57	89.44±0.21	67.44±0.87	58.19±0.93	**98.60**±0.10	**95.87**±0.38	**98.61**±0.10	75.30±0.87	64.97±1.98	87.99±0.55
– Dim (128)	73.82±1.47	63.63±2.33	72.82±1.51	89.80±0.03	66.99±1.97	58.03±0.95	98.37±0.20	94.66±0.60	98.44±0.15	74.77±0.63	65.10±1.62	87.85±0.36
– k_m (down)	71.63±1.41	62.33±2.40	70.45±1.48	90.17±0.09	67.65±0.96	59.51±0.60	98.08±0.12	95.40±0.43	98.07±0.11	75.56±0.14	66.57±0.50	88.53±0.06
– k_m (up)	73.66±2.14	64.21±2.01	**72.98**±2.20	89.68±0.22	66.82±0.80	58.88±0.82	98.05±0.39	93.46±1.26	98.06±0.38	74.57±0.36	65.39±0.82	87.93±0.24
– $S \downarrow$	73.51±0.99	64.49±1.74	73.11±1.23	91.12±0.25	69.03±0.45	60.10±0.50	97.73±0.16	93.61±1.88	98.12±0.57	75.38±0.32	66.78±0.34	87.49±0.09
– $S \uparrow$	73.06±0.78	63.22±2.04	71.84±1.13	89.19±0.09	66.73±1.30	58.20±1.33	97.51±0.16	92.88±1.33	97.50±0.18	75.53±0.48	66.37±0.33	86.80±0.12
– Linear $\hat{\phi}(\mathbf{w})$	**74.09**±0.76	**64.53**±1.35	72.94±1.05	**91.63**±0.22	**71.22**±1.31	60.84±0.08	98.08±0.41	93.96±0.59	98.07±0.41	**76.37**±0.84	**67.31**±0.35	**89.10**±0.20

As in Tables 8–10, our method is not sensitive to the hyperparameters. Changing the token embedding dimension from 64 to 32 or 128 preserves competitive performance. Similarly, changing the stride by one step in either direction, $S \downarrow$ or $S \uparrow$, often maintains performance close to the main configuration. This indicates that the proposed tokenizer reduces sensitivity not only to the token dimension, but also to the parameters used to form tokens.

The filter-length ablations show a similar trend. In some datasets, modifying the filter lengths improves performance, while in others the main configuration remains stronger. However, we fixed the same set of lengths within each task rather than tuning them separately for each dataset, in order to better assess the generalization of the method.

In some datasets, replacing the convolutional mapping with an explicit linear projection improves performance, especially on ECG and EEG datasets (Table 10). This suggests that the convolutional inductive bias is not uniformly better across tasks. For some datasets, a linear window-to-token mapping may provide a more suitable representation.

We also conducted additional ablation studies on larger token dimensions. During experiments, increasing the token embedding dimension up to 1024 did not lead to consistent performance improvements. This further suggests that the gains in our method come from constructing more stable token representations through multiple learned views and summation.

Temporal Support and Kernel Robustness. Because the same kernel length corresponds to different temporal support at different sampling rates, Table 11 reports the kernels used and their duration in seconds. This also shows that our method is not tied to a single absolute size. Instead, it remains robust across modalities by using a small set of supports matched to the tasks.

Table 11: k_m **and temporal support.** Kernel lengths k_m are reported in samples, and their temporal support is k_m/f_s seconds. Across sampling rates and modalities, the tokenizer uses a small set of supports, showing that the method remains robust without dataset-specific tuning of many choices.

Modality / datasets	Sampling rate f_s	Kernel lengths k_m	Temporal support
PPG	25 Hz	$\{9, 17, 33\}$	$\{0.36, 0.68, 1.32\}$ s
IMU activity	50 Hz	$\{5, 9\}$	$\{0.10, 0.18\}$ s
Clemson step counting	15 Hz	$\{5, 9\}$	$\{0.33, 0.60\}$ s
ECG / EEG	100 Hz	$\{33, 65, 129\}$	$\{0.33, 0.65, 1.29\}$ s

E.1 Gating Weights

We also investigated the learned gate parameters g_m for each k_m. We reported the results below.

Table 12: Average learned gating weights α_m across datasets for PPG and IMU modalities. Values are reported as mean $\pm$ standard deviation across runs.

(a) PPG datasets

k_m	IEEE SPC	WildPPG	DaLiA	BIDMC
9	0.55 ± 0.03	0.54 ± 0.01	0.53 ± 0.02	0.54 ± 0.01
17	0.48 ± 0.02	0.48 ± 0.02	0.50 ± 0.01	0.45 ± 0.03
33	0.49 ± 0.02	0.48 ± 0.02	0.48 ± 0.02	0.51 ± 0.03

(b) IMU datasets

k_m	UCIHAR	HHAR	USC	Clemson
5	0.54 ± 0.03	0.53 ± 0.02	0.53 ± 0.02	0.44 ± 0.03
9	0.50 ± 0.02	0.48 ± 0.02	0.49 ± 0.01	0.56 ± 0.03

Table 13: Average learned gating weights α_m for the multitaper tokenizer on the ECG and EEG datasets. Values are reported as mean $\pm$ standard deviation across runs.

k_m	PhysioNet2017	PTB-XL	Chapman	Sleep-EDFx
33	0.54 ± 0.01	0.49 ± 0.02	0.53 ± 0.01	0.45 ± 0.02
65	0.48 ± 0.02	0.51 ± 0.02	0.49 ± 0.01	0.48 ± 0.02
129	0.45 ± 0.02	0.46 ± 0.01	0.46 ± 0.02	0.59 ± 0.01

The learned gates are generally stable across datasets and remain close to 0.5. They are also similar within each task, suggesting that the relative contribution of the views is not dataset-specific. This indicates that the different temporal views contribute in a balanced way in most cases, which is consistent with the intuition behind traditional multitaper methods, where multiple views are combined with comparable weights. An exception is Sleep-EDFx, where the gates show clearer separation. In this case, the model assigns more weight to the longer segment (0.59). Importantly, the main-text ablations in Section 6.1 show that using the same fixed weights across views still gives strong performance, further supporting that the gains do not come from gate alone.

E.2 Significance Analysis

Overall, our method achieves the best performance on 10 of the 12 datasets and remains competitive on the other two. To test whether this is statistically robust, we perform a rank-based significance analysis across datasets.

Because multiple metrics reported for the same dataset can be correlated, we first conduct a dataset-level analysis using one representative metric per dataset. For heart-rate and step counting estimation, we use correlation ρ; for activity, we use weighted F1; for PTB-XL, we use AUROC; for Chapman, we use weighted F1; and for sleep staging, we use Cohen's κ. This gives one evaluation metric per dataset, avoiding the assumption that multiple metrics from the same dataset are independent.

For each dataset, we rank all methods over 3 random seeds using the selected metric, with lower rank indicating better performance and ties assigned averaged ranks. We then average ranks across seeds and apply the Friedman test across the 12 dataset-level tasks, followed by Nemenyi post-hoc comparisons.

Figure 6 shows that Multitaper achieves the best average rank under this stricter dataset-level evaluation. This indicates that the advantage of the proposed tokenizer is not driven by repeatedly counting correlated metrics from the same dataset, but remains visible when each dataset contributes only one representative outcome.

For completeness, we also report an auxiliary analysis over all reported metrics, treating each dataset–metric pair as a separate task. This gives 36 tasks in total (12 datasets $\times$ 3 metrics). Although this analysis should be interpreted as a sensitivity check because metrics from the same dataset are not fully independent, it provides a broader view across all reported performance measures.

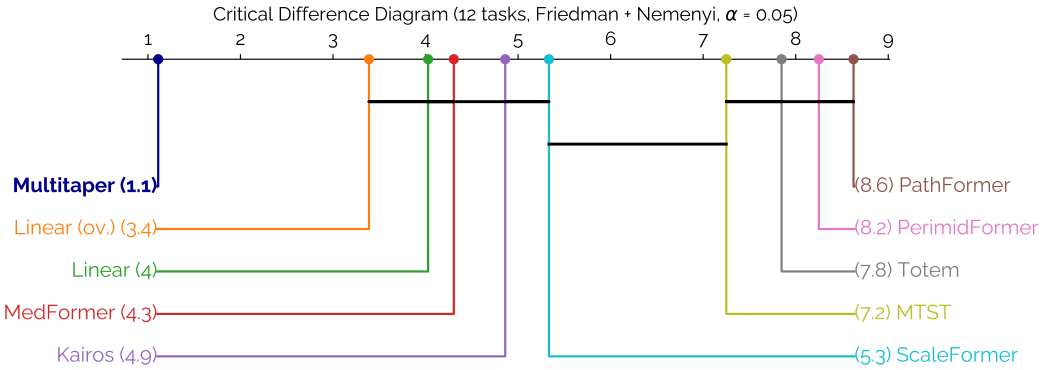

Figure 6: Critical-difference diagram from the Friedman test with Nemenyi post-hoc comparisons using one representative metric per dataset. Lower average rank is better. This dataset-level analysis avoids treating correlated metrics from the same dataset as independent.

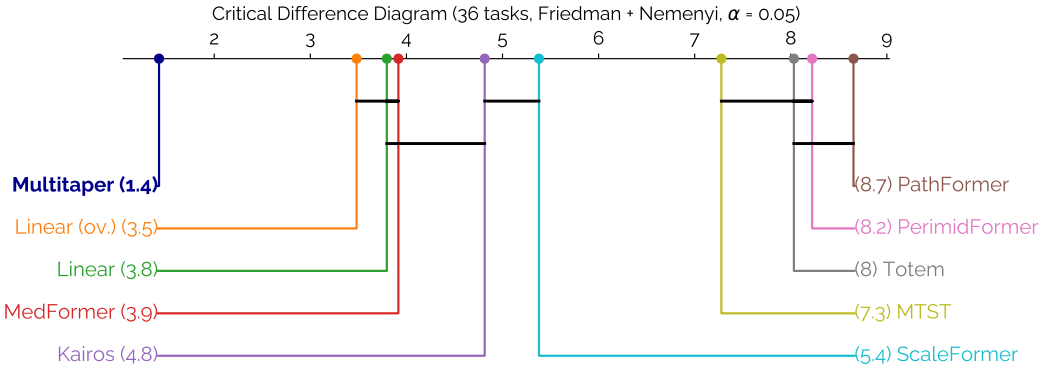

Figure 7: Critical-difference diagram from the Friedman test with Nemenyi post-hoc comparisons over all 36 dataset–metric pairs. Lower average rank is better. This analysis is reported as a complementary sensitivity check because metrics from the same dataset can be correlated.

The all-metric analysis leads to the same qualitative conclusion. Our method obtains the best average rank across tasks. Together, the dataset-level and all-metric analyses show that our method delivers the strongest overall performance profile across datasets and metrics.

F Self-Supervised Learning Experiments

Since Transformer architectures are widely used for self-supervised learning (SSL), we have also conducted additional experiments using our proposed tokenizer integrated transformer model as a backbone and compared it with traditional Transformer model.

For SSL experiments, we selected Learning Without Augmenting [119], a recent state-of-the-art SSL method for temporal sequences. Unlike augmentation-based approaches [120, 121], it avoids hand-designed augmentations and therefore reduces augmentation-induced bias. This choice is also preferable to masking-based Transformer pretraining in our setting, because masking methods require task-specific masking ratios and additional decoders, while Learning Without Augmenting has shown stronger performance than masking-based temporal SSL methods. Moreover, since this method can be implemented with any encoder architecture rather than requiring a specialized model design [122], it allows us to directly integrate our tokenizer into the original Transformer backbone.

We followed the original experimental setup with a plain Transformer encoder and our tokenizer integrated. We also included the ResNet encoder used in the original Learning Without Augmenting implementation, which is an eight-block 1D residual convolutional encoder [123]. To ensure a fair comparison, we chose comparable channel widths and depts so that the trainable parameter counts remained similar across models. For datasets that were not included in the original paper, we used a larger subject-independent subset for pretraining and then fine-tuned on a much smaller training subset, following prior SSL methods on temporal sequences [109, 124]. In all cases, we enforced subject-independent splits, so that subjects used in pretraining or fine-tuning were never included in the test set. Tables 14 to 16 present the resulting comparisons.

Table 14: Performance comparison of methods for SSL task in HR estimation using PPG.

Method	IEEE SPC			WildPPG			DaLiA			BIDMC		
	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑	MAE ↓	RMSE ↓	ρ ↑
ResNet	8.84±0.50	14.37±0.95	82.67±1.30	**8.62**±0.06	**15.00**±0.06	42.41±0.24	10.04±0.33	18.26±0.43	60.16±0.06	2.41±0.06	3.95±0.10	87.58±0.09
Plain	10.22±0.28	16.05±0.47	72.84±1.94	9.92±0.04	17.82±0.03	41.50±0.27	6.18±0.13	12.67±0.21	72.30±0.48	1.92±0.03	4.42±0.08	84.96±0.09
Multitaper	**8.09**±0.24	**13.12**±0.41	**89.76**±0.88	9.03±0.04	15.98±0.05	**45.32**±0.21	**5.08**±0.09	**10.92**±0.11	**84.71**±0.16	**1.59**±0.05	**3.63**±0.12	**92.84**±0.74

Table 15: Performance comparison of methods for SSL task in activity and step using IMU.

Method	UCIHAR			HHAR			USC			Clemson		
	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	Acc ↑	F1 ↑	W-F1 ↑	MAPE ↓	MAE ↓	RMSE ↓
ResNet	77.55±1.10	78.50±0.76	81.17±0.55	70.67±0.06	67.74±0.29	68.79±0.25	52.21±1.09	48.64±1.52	48.22±1.11	5.16±0.44	2.50±0.13	4.65±0.45
Plain	68.92±0.84	69.41±0.61	71.85±0.66	61.48±0.11	59.37±0.34	60.02±0.29	45.17±1.41	42.58±1.77	41.96±1.43	5.92±0.31	2.87±0.14	**4.12**±0.31
Multitaper	**80.11**±1.36	**80.22**±1.04	**83.94**±0.83	**76.81**±0.18	**73.82**±0.40	**74.97**±0.31	**56.87**±1.18	**53.12**±1.44	**52.46**±1.22	**4.63**±0.28	**2.29**±0.10	5.05±0.37

Table 16: Performance comparison of methods for SSL task in CVD and sleep using ECG and EEG.

Method	PhysioNet 2017			PTB-XL			Chapman			Sleep		
	Acc ↑	F1 ↑	W-F1 ↑	AUROC ↑	F1 ↑	Acc ↑	Acc ↑	F1 ↑	W-F1 ↑	κ ↑	F1 ↑	W-F1 ↑
ResNet	65.12±0.30	52.87±1.55	66.91±0.61	88.03±0.45	66.12±0.51	58.21±1.20	87.21±0.80	85.30±0.98	93.10±1.13	69.16±1.32	58.32±0.83	68.05±0.86
Plain	60.84±0.41	44.77±1.12	58.94±0.73	87.46±0.37	63.29±0.64	54.12±1.03	78.36±0.44	76.84±0.87	84.63±0.91	63.02±0.72	53.48±0.66	79.05±0.25
Multitaper	**70.18**±0.99	**58.05**±1.48	**69.47**±0.52	**90.13**±0.10	**72.18**±0.41	**62.02**±0.74	**90.06**±0.83	**88.13**±0.91	**94.17**±0.21	**73.12**±1.04	**60.21**±1.03	**82.30**±0.43

Overall, the proposed tokenizer improves the Transformer backbone while even outperforming the ResNet baseline despite the strong architectural prior and inductive bias built into convolutional designs. ResNet already benefits from locality and weight sharing at the architecture level, whereas our gains come from improving the tokenization stage while leaving the Transformer backbone unchanged. Moreover, these comparisons are obtained without data augmentation, so they reflect the plain representation-learning ability of each model rather than gains introduced by augmentation-specific biases.

G Time-Series Forecasting

We evaluate our tokenizer on long-term forecasting to examine its use beyond health-related tasks and attention-based backbones. We integrate it into TimePro [125], a Mamba-based forecasting model, by replacing only its patch embedding. We retain TimePro's backbone, stride, token count, and embedding dimension. Each view uses only the observed context $x_{1:L}$. Following TimePro, we use a context length $L = 96$ and report MSE and MAE averaged over prediction horizons $H \in \{96, 192, 336, 720\}$. We evaluate Exchange, Weather, ETTm1, and ETTh1, and include S-Mamba [126] as an additional reference.

We first use the same three receptive-field lengths, $k_m \in \{5, 9, 17\}$, across all datasets and horizons. Since these datasets have different sampling intervals, the shared bank covers different physical durations. We then select view banks on validation data through a dyadic grid search. For ETTm1 and ETTh1, validation selects the shared bank for all horizons. For Weather, it selects $\{3, 5, 9\}$ for $H \in \{96, 192, 336\}$ and $\{5, 9, 17\}$ for $H = 720$. For Exchange, it selects $\{9, 17, 33\}$ for $H = 336$ and $\{5, 9, 17\}$ for the remaining horizons. Stride, token count, and embedding dimension remain unchanged during this selection.

For TimePro and the validation-selected tokenizer, Table 17 reports mean and standard deviation across seeds 2023, 2024, and 2025 on Exchange, Weather, and ETTh1. For each seed, we first average each metric over the four prediction horizons. Our tokenizer retains lower mean errors across the three-seed comparisons.

Table 17: Forecasting errors averaged over four prediction horizons.

Method	**Metric**	**Exchange**	**Weather**	**ETTm1**	**ETTh1**
S-Mamba	MSE	0.367	0.251	0.398	0.455
	MAE	0.408	0.276	0.405	0.450
TimePro	MSE	0.352 ± 0.01	0.251 ± 0.02	0.391 ± 0.02	0.438 ± 0.01
	MAE	0.399 ± 0.03	0.276 ± 0.03	0.400 ± 0.02	0.438 ± 0.02
TimePro + Ours (shared)	MSE	0.347 ± 0.02	0.248 ± 0.03	$\mathbf{0.385} \pm 0.01$	$\mathbf{0.431} \pm 0.02$
	MAE	0.390 ± 0.01	0.273 ± 0.02	$\mathbf{0.393} \pm 0.02$	$\mathbf{0.432} \pm 0.02$
TimePro + Ours (selected)	MSE	$\mathbf{0.343} \pm 0.02$	$\mathbf{0.245} \pm 0.02$	$\mathbf{0.385} \pm 0.02$	$\mathbf{0.431} \pm 0.02$
	MAE	$\mathbf{0.387} \pm 0.02$	$\mathbf{0.270} \pm 0.03$	$\mathbf{0.393} \pm 0.02$	$\mathbf{0.432} \pm 0.02$

Our tokenizer sums the temporal views before they enter the forecasting backbone, so each position still produces a single token. The backbone receives the same number of tokens with the same embedding dimension, and all layers after tokenization remain unchanged. These results therefore demonstrate improvements through token formation alone, without modifying the subsequent forecasting architecture.

H Computational Overhead Comparison

Our tokenizer adds a small overhead to the base model. First, unlike methods that treat different channels or variates as separate tokens, our method keeps the original temporal token grid and sums multiple views before the Transformer. Therefore, it does not increase the token count, which is important because the cost of self-attention grows quadratically with the sequence length.

Second, the tokenization can be implemented as a single bank. All K views are computed in parallel and then summed by scalar gates. Thus, the computation is confined to the tokenizer and scales linearly with the number of views K, while the Transformer architecture remains unchanged.

Table 18: **Computational overhead of tokenization strategies.** Comparison of tokenizer parameter complexity and the token sequence length passed to the Transformer backbone. Here, L is the input length, C_{in} is the number of channels, R is the patch size or receptive field, S is the temporal stride, d is the embedding dimension, and K is the number of views or scales. For our method, k_j denotes the kernel size of view j.

Method	Tokenization Pattern	Channel Handling	Tokenizer Parameters	Transformer Token Length
Linear (non-overlapping)	Non-overlap patching	Independent	$\mathcal{O}(Rd)$	$C_{\text{in}} \cdot \frac{L}{R}$
Linear (overlapping)	Overlap patching	Independent	$\mathcal{O}(Rd)$	$C_{\text{in}} \cdot \frac{L}{S}$
Naive multi-view concat	Overlapping multi-view	Independent	$\mathcal{O}(KRd)$	$K \cdot C_{\text{in}} \cdot \frac{L}{S}$
MTST [56]	Multi-scale patching	Joint / branch-fused	$\mathcal{O}\left(d \sum_{i=1}^{K} R_i\right)$	$C_{\text{in}} \cdot \sum_{i=1}^{K} \frac{L}{S_i}$
MedFormer [54]	Multi-scale patching	Joint / fused	$\mathcal{O}\left(C_{\text{in}} d \sum_{i=1}^{K} R_i\right)$	$\sum_{i=1}^{K} \frac{L}{R_i}$
Ours	Overlapping multi-view	Joint / fused	$\mathcal{O}\left(C_{\text{in}} d \sum_{j=1}^{K} k_j + K\right)$	$\frac{L}{S}$

Our method is the only multi-view tokenizer in the comparison that keeps the Transformer token number unchanged, which is important because the downstream self-attention cost scales quadratically with the length. Moreover, for architecture-coupled baselines such as MedFormer, this table summarizes the tokenization front-end rather than the full complexity of the attention.

The parameter overhead is also small. The additional parameters consist primarily of the convolution bank weights and K scalar gates. More precisely, for embedding dimension d, channel count C, and view-specific kernel sizes $\{k_j\}_{j=1}^{K}$, the tokenizer introduces

$$C_{\text{in}} d \sum_{j=1}^{K} k_j + K$$

parameters. This is modest compared to approaches that introduce separate routing modules, cross-scale attention blocks, or models based on VQ-VAE encoders and codebooks. In particular, our method does not require an additional reconstruction model or codebook learning.

From the computation side, the added cost is the tokenizer and scales linearly with the number and size of the kernels. In our experiments, the heaviest three-filter configuration added less than 0.1 ms to inference time on an NVIDIA RTX 4090.

We also note that the token count itself directly affects computational cost, since the complexity of self-attention grows quadratically with the number of tokens. To make the comparison fair, we selected the stride for each task from the candidate set $S = \{2, 4, 8, 16, 32\}$ using the best-performing setting for that task. Even under this tuning for the baselines, our method remained among the most efficient approaches in terms of the number of tokens passed to the Transformer. This is because our method improves the tokenization stage without duplicating token streams across views, so the token sequence length seen by the backbone stays fixed.

Model size comparison. The complexity expressions in Table 18 describe how each tokenization strategy scales with input length, channel count, and token length. However, several baselines, including PathFormer, MedFormer, ScaleFormer, MTST, Kairos, and TOTEM, modify more than the tokenization layer. Therefore, we also report representative trainable parameter counts on three datasets. This comparison captures the practical model-size overhead introduced by each method, including tokenizer parameters, routing modules, scale-specific blocks, reconstruction/codebook components, and architecture-specific layers.

Table 19: **Representative trainable parameter counts.** We report total trainable parameters for three representative datasets chosen to cover different sequence lengths and channel counts, using the same experimental settings as the main results. Parameter counts are rounded to the nearest thousand. For TOTEM, "$a + b$" denotes the model parameters plus the VQ-VAE tokenizer parameters. We also indicate whether each method changes the Transformer backbone or only modifies the tokenizer.

Method	Backbone changed	Extra mechanism	DaLiA	USC	PTB-XL
Linear	No	Standard patch embedding	211k	217k	201k
Linear (ov.)	No	Overlapping patch embedding	211k	214k	202k
Single-kernel	No	Single convolutional tokenizer	212k	203k	310k
TOTEM [51]	No	VQ-VAE/codebook tokenizer	320k+400k	380k+420k	500k+300k
PathFormer [53]	Yes	Multi-scale paths/routing	450k	1M	600k
MedFormer [54]	Yes	Multi-scale patching/fusion	300k	500k	411k
ScaleFormer [55]	Yes	Scale-based modules	255k	256k	300k
MTST [56]	Yes	Multi-scale patching	600k	700k	1.2M
Kairos [57]	Yes	Adaptive segmentation/routing	490k	500k	495k
Ours	**No**	**Gated summation of learned views**	213k	206k	349k

Table 19 complements the asymptotic comparison by reporting the actual trainable parameter counts used in our experiments. Our method keeps the added capacity localized to the tokenizer and does not introduce additional blocks. The parameter counts of the linear and overlapping-linear tokenizers can vary across datasets because their best-performing window or stride settings are not always the same.

Also, the apparent inversion between Linear and Ours is due to how the tokenizer parameters scale with the dataset-specific input shape. Although the Transformer encoder backbone is kept fixed within each dataset, the input tokenizer is different. For a linear mapping, the number of tokenizer parameters scales as $O(d\, M_\Delta)$, because separate channels can use the same mapping. In contrast, our multi-kernel tokenizer scales approximately as $O\left(d\, C_{\text{in}} \sum_{j=1}^{K} k_j\right)$, because it applies views with sizes k_j to all channels. Therefore, the relative parameter count depends on the channel count C_{in}, the selected window size M_Δ, and the bank. For PTB-XL, the input has 12 channels, and the multi-kernel bank increases the tokenizer size, so Ours has more parameters than Linear.

The larger counts for MedFormer and MTST come from a different source. Both methods use multiple resolutions and process the resulting branches through additional scale specific modules. MedFormer uses multi-granularity embedding and two-stage intra-/inter-granularity attention, while MTST uses a multi-branch architecture for different temporal resolutions. As a result, their parameter counts can grow quickly when several window branches are used.

We also attempted to reduce the parameter counts of the larger baselines by decreasing their hidden dimensions, numbers of layers, or scale-specific components. However, these smaller configurations did not improve performance in our experiments and often reduced accuracy. Therefore, we report the parameter counts corresponding to the best-performing settings used in the main comparisons rather than smaller variants.

Across the representative datasets, this comparison shows that our method improves tokenization while keeping the backbone unchanged and the parameter overhead small. Overall, it provides a lightweight alternative to prior methods with substantially more complex routing or fusion.

Practical token counts. To complement the asymptotic comparison, Table 20 reports the effective number of tokens passed to the Transformer using the selected hyperparameters from Table 7 with maximum length samples from datasets. These settings are the tuned parameters used in the main experiments, so the comparison reflects the practical token counts of the evaluated models.

For the non-overlapping Linear (PatchTST) baseline, the stride equals the window length, so the count is $C_{\text{in}}\lceil L/M_\Delta \rceil$. For Linear (ov.), the count is $C_{\text{in}}\lceil L/S \rceil$. For multitaper, the bank fuses channels before the Transformer, so the count is only $\lceil L/S \rceil$. Thus, the token-count advantage is strongest for multichannel datasets, where independent channel tokenization multiplies the sequence length

by C_{in}. Despite the tuning of Linear and Linear (ov.) for the evaluated settings, their performance remains weaker in many datasets, indicating that the gains are not simply due to using more tokens.

Table 20: **Effective token counts passed to the Transformer.** For Linear, the stride is $S = M_{\Delta}$. Linear (ov.) uses the selected overlapping stride S. Multitaper uses multiple kernel views but sums them before the Transformer, so its token count depends only on the temporal stride and not on C_{in}.

Dataset	C_{in}	L	Linear M_{Δ}/S	Linear tokens	Linear (ov.) M_{Δ}/S	Linear (ov.) tokens	Multitaper S	Multitaper tokens
IEEE SPC	1	200	$32/32$	7	$16/4$	50	4	50
DaLiA	1	200	$4/4$	50	$8/4$	50	4	50
WildPPG	1	200	$8/8$	25	$8/4$	50	4	50
BIDMC	1	200	$64/64$	4	$8/4$	50	4	50
UCIHAR	9	128	$16/16$	72	$8/4$	288	2	64
HHAR	6	100	$2/2$	300	$8/4$	150	2	50
USC	6	100	$16/16$	42	$8/4$	150	2	50
Clemson	1	480	$8/8$	60	$64/32$	15	2	240
PhysioNet2017	1	9000	$128/128$	71	$128/16$	563	8	1125
PTB-XL	12	1000	$4/4$	3000	$8/4$	3000	8	125
Chapman	4	1000	$32/32$	128	$8/4$	1000	8	125
Sleep-EDFx	1	3000	$64/64$	47	$32/8$	375	8	375

The calculation highlights the main efficiency difference. For example, PTB-XL has 12 channels $C_{\text{in}} = 12$ and $L = 1000$. Linear (ov.) with $S = 4$ passes $12\lceil 1000/4 \rceil = 3000$ tokens to the Transformer, whereas multitaper with $S = 8$ passes only $\lceil 1000/8 \rceil = 125$ tokens. Similarly, UCIHAR has $C_{\text{in}} = 9$, so Linear (ov.) passes $9\lceil 128/4 \rceil = 288$ tokens, while multitaper passes $\lceil 128/2 \rceil = 64$. Therefore, our method is especially efficient in multichannel settings because it improves the tokenizer with multiple views without duplicating the token stream seen by self-attention.

We also experimented with increasing the token count of the linear baselines by using smaller patch lengths or strides. However, the best performance was obtained with the hyperparameters reported in Table 7. This indicates that simply passing more tokens to the Transformer does not necessarily improve performance; the tokenizer must also produce useful estimates.

I Motivating Example and Bound

In this section, we first provide additional evidence of the effect of windowing on tokenization across all datasets and metrics. We then revisit this effect through the lens of the developed theory. Finally, we fit a curve inspired by the bound to the observed performance across window sizes.

I.1 Further Evidence with Other Tasks

To complement the motivating example in the main text, we report the effect of window size across all datasets and tasks. Whereas Figure 1 shows the relative degradation from the best window for the HR datasets, here we provide the full results in absolute metric values, which are possible to include in the Appendix due to space constraints in the main text.

Figures 8–10 show that the dependence on window size is not unique to HR estimation. Across PPG, IMU, ECG, and EEG tasks, performance changes with the tokenization window, and the best choice is not consistent across datasets, even within the same domain. These additional results show that the optimal tokenization window depends on the dataset and task.

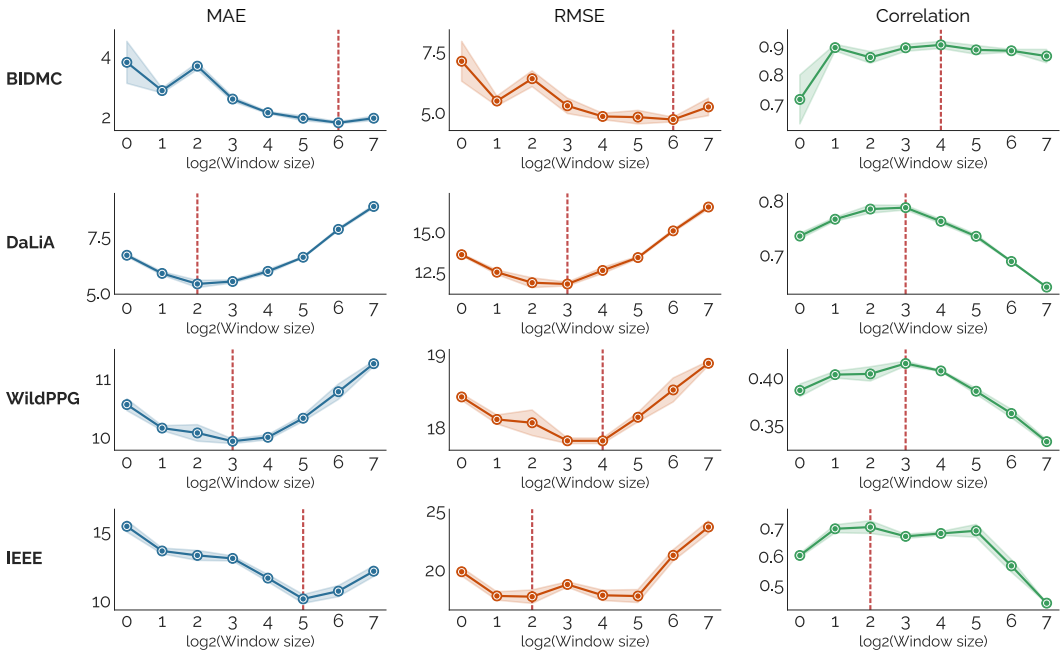

Figure 8: **Window-size sensitivity for heart-rate estimation.** Performance is shown across tokenization window sizes, where the x-axis reports $\log_2(M_\Delta)$ and M_Δ is the number of samples per window. Each row corresponds to a dataset and each column to an evaluation metric. The red dashed line marks the best-performing window size for each dataset and metric.

Overall, the effect of window size is often substantial. In several datasets, using a non-optimal window leads to clear degradation, with drops reaching 46.44% on PhysioNet 2017, and 39.14% on PTB-XL. At the same time, the sensitivity is not uniform. Some datasets such as UCIHAR, HHAR, and USC are more stable compared to other tasks, with changes mostly around 3–13%. We also observe that agreement across metrics is dataset dependent. For some datasets, all metrics select the same best window (e.g., PTB-XL, Sleep-EDFx, UCIHAR, and USC), whereas for others the best window varies across metrics. This further supports the main point that the choice of window size is data dependent and affect performance.

In the next section, we will relate these performance changes to the theory using statistical features extracted from datasets for heart rate estimation.

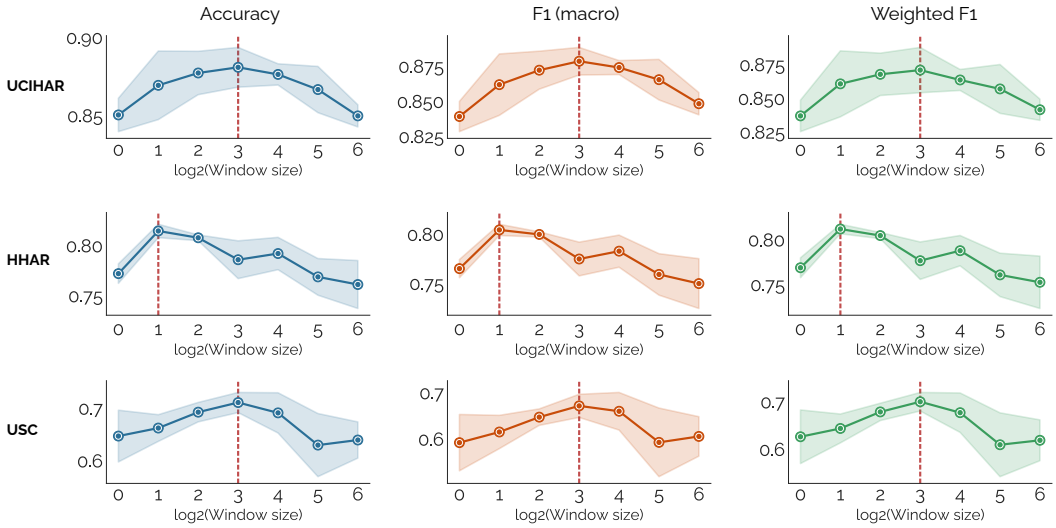

Figure 9: Window size comparisons in metrics for all datasets in activity detections task

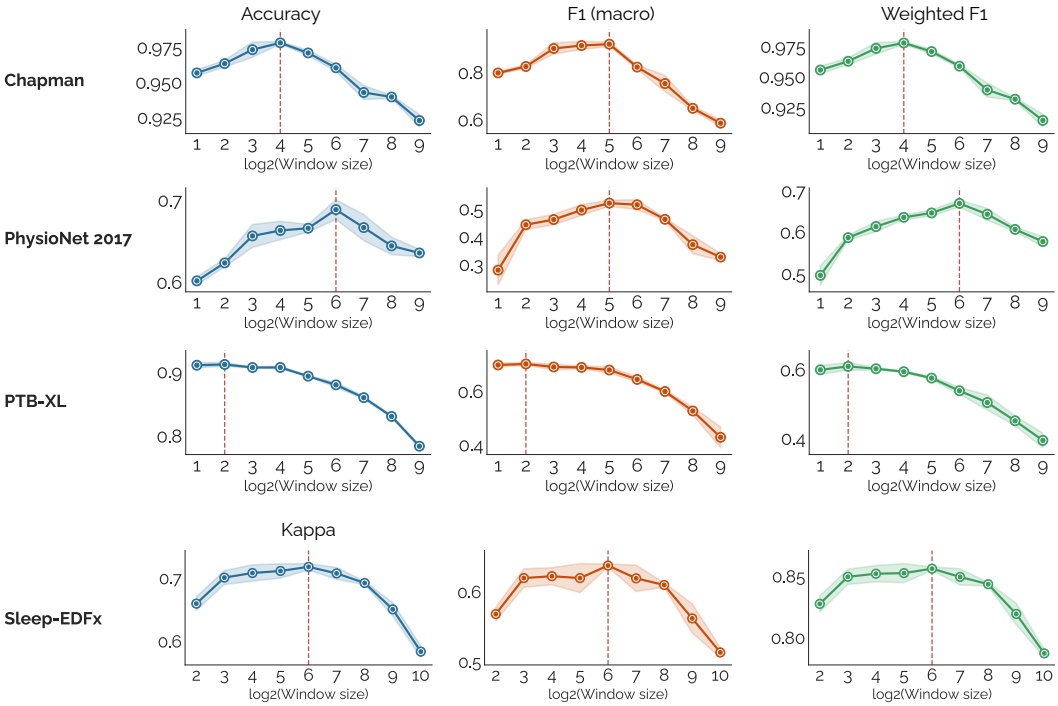

Figure 10: Window size comparisons in metrics for all datasets in cardiovascular diseases and sleep stage classification task

I.2 Revisiting the Motivational Example

To connect the theory to real-world data, we revisit heart rate estimation from photoplethysmography signals on three representative datasets. The preferred tokenization window size M_Δ varies substantially across datasets. BIDMC achieves its best performance at $\log_2(M_\Delta) = 6$, i.e., $M_\Delta^\star = 64$, whereas DaLiA and WildPPG prefer smaller windows, with $M_\Delta^\star = 4$ and $M_\Delta^\star = 8$, respectively.

For uniform pooling with uncorrelated residuals, Equation 6 has the following form with dataset-specific coefficients.

$$\mathbb{E}\left[\|\hat{\theta} - \theta^\star\|^2\right] \leq A_d \Delta^{2r} + B_d \Delta^{-(p-1)}.$$

We use spectral entropy (SE) and neighbor Jensen–Shannon divergence (JSD) to describe local variation, and signal-to-noise ratio (SNR) to describe signal quality. Higher SE and JSD suggest more variation for each token to summarize, while lower SNR can make local estimation less reliable. These statistics provide qualitative context for the two terms without specifying a numerical relation to A_d and B_d.

The corresponding dataset summaries, written as $(\overline{\mathrm{SNR}}, \overline{\mathrm{SE}}, \overline{\mathrm{JSD}}, M_\Delta^\star)$ are

$$(3.13,\ 0.38,\ 0.028,\ 64)_{\mathrm{BIDMC}}, \quad (-5.40,\ 0.63,\ 0.088,\ 4)_{\mathrm{DaLiA}}, \quad (-7.09,\ 0.55,\ 0.080,\ 8)_{\mathrm{WildPPG}}.$$

BIDMC has the highest SNR and the lowest variability proxies, so both the estimation burden and the approximation burden are relatively mild. This allows the model to benefit from a large window, where more samples are available per token and the variance term is reduced without substantial approximation loss. By contrast, DaLiA and WildPPG exhibit lower SNR together with higher SE and JSD, which indicate noisier and more heterogeneous local structure. In these datasets, increasing M_Δ improves sample support but also forces a single token to summarize a more variable segment, so the approximation term grows quickly.

Thus, the motivating example is consistent with the theory. Datasets with cleaner and more stationary structure favor larger windows, whereas datasets with noisier and rapidly varying local structure favor smaller windows.

I.3 Empirical Fit of the Theoretical Window-Size

To move beyond a qualitative comparison, we fit the theorem-inspired form of the window-size tradeoff to the observed performance curves. For each dataset $d \in \{\mathrm{BIDMC}, \mathrm{DaLiA}, \mathrm{WildPPG}\}$, we convert the performance curve over window sizes into an error curve $\mathcal{L}_d(M_\Delta)$, where lower values are better. We then fit the dataset-specific surrogate as in Equation 11.

$$\hat{\mathcal{L}}_d(M_\Delta) = \alpha_d M_\Delta^{\eta} + \beta_d M_\Delta^{-\gamma}, \tag{11}$$

The surrogate uses the increasing and decreasing terms of the uniform, uncorrelated bound in Equation 6 to describe the empirical error curve. Since $M_\Delta = f_s\Delta$, the conversion to sample counts is absorbed into the coefficients. The coefficients α_d and β_d are dataset-specific, while η and γ describe the growth and decay of the fitted terms.

The optimal window size predicted by the fitted curve is obtained by differentiating Equation 11 and setting the derivative to zero, where we obtain the expression in Equation 12.

$$M_{\Delta,d}^\star = \left(\frac{\gamma\,\beta_d}{\eta\,\alpha_d}\right)^{\frac{1}{\eta+\gamma}} \tag{12}$$

Thus, the predicted optimum depends on the ratio between the estimation coefficient β_d and the approximation coefficient α_d, exactly as suggested by the theory.

To reduce the risk of overfitting, we do not fit an unconstrained four-parameter curve independently to each dataset. Instead, we use a shared-shape surrogate in which the exponents η and γ are shared across datasets, while only the scale coefficients α_d and β_d are dataset-specific.

Therefore, the fit constrains all datasets to follow the same approximation–estimation tradeoff principle, while allowing each dataset to differ in the relative strength of the two terms. This is consistent with the theory, where the exponents describe how the two error components scale with window size, while the dataset-specific coefficients absorb properties of the data and task that affect the strength of each component. For example, quantities such as SNR, spectral entropy, and neighbor JSD are not modeled as separate variables in Equation 11; instead, their effects are reflected in the fitted coefficients α_d and β_d. Thus, α_d can be interpreted as capturing the dataset-specific approximation burden, while β_d captures the dataset-specific estimation burden.

Figure 11 shows the resulting fits. The colored curves are the observed values, while the black curves are the fitted theory-derived surrogates. Across all datasets, the fitted curves reproduce the qualitative structure of the empirical tradeoff.

The fitted coefficients are also consistent with the dataset statistics used earlier as empirical proxies. BIDMC has the lowest spectral entropy and neighbor JSD, which corresponds to a smaller approximation burden and therefore a weaker growth of the first term in Equation 11. This shifts the optimum

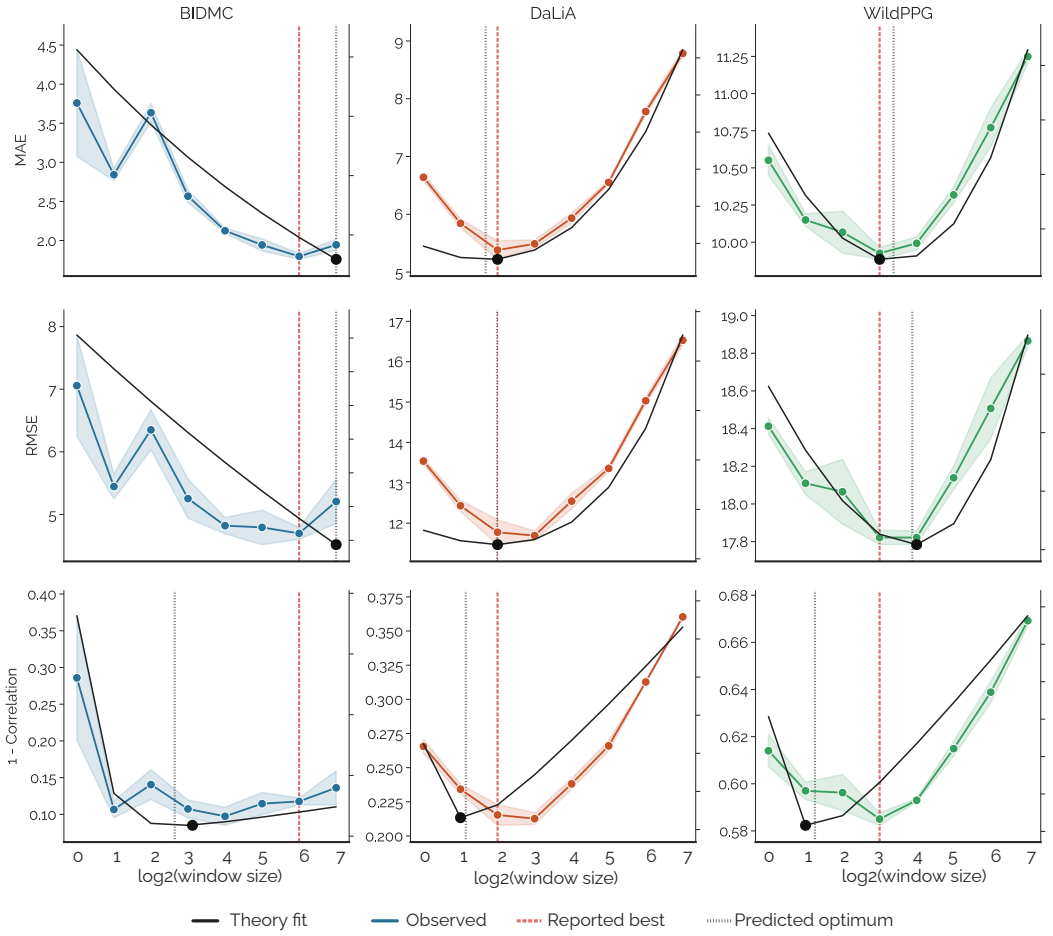

Figure 11: **Direct empirical fit of the theorem window-size** The colored curve shows the observed normalized metrics as a function of $\log_2(M_\Delta)$, and the black curve shows the fitted surrogate $\alpha_d M_\Delta^{\eta} + \beta_d M_\Delta^{-\gamma}$. The red dashed line marks the window selected by minimum observed MAE, and the gray dotted line marks the optimum predicted by the fitted theory curve.

to the right and favors a larger M_Δ. By contrast, DaLiA and WildPPG exhibit substantially larger variability proxies, so the approximation term grows more quickly with window size and the fitted optimum shifts to smaller windows. At the same time, their poorer SNR increases the estimation burden, so the observed optimum reflects the balance of both terms rather than either one alone.

The fit is also not expected to be exact. The theorem provides an upper-bound-inspired decomposition rather than a generative law for the empirical curves, and the dataset-level quantities $\overline{\mathrm{SNR}}$, $\overline{\mathrm{SE}}$, $\overline{\mathrm{JSD}}$ are only approximates for the latent constants. Nevertheless, the fitted surrogate captures the main phenomenon well. This agreement supports the approximation–estimation interpretation of the observed window-size effects.

J Further Experiments with Synthetic Data

To directly evaluate the developed tradeoff, we construct a synthetic benchmark. We use the original Transformer backbone with the sinusoidal positional embeddings, and vary the tokenization window length in different tasks. This allows us to isolate how windowing affects performance.

J.1 Experimental Protocol

We evaluate three Transformer encoder sizes in the synthetic experiments. The *small* model uses token dimension $d = 96$, depth $L = 3$, $H = 2$ attention heads, and feedforward dimension 192. The *medium* model uses $d = 128$, $L = 4$, $H = 4$, and feedforward dimension 256. The *large* model uses $d = 192$, $L = 6$, $H = 6$, and feedforward dimension 384. All models use dropout 0.1. For each model, we evaluate a family of tokenization window lengths

$$M_\Delta \in \{1, 2, 4, 8, 16, 32, 64, 128, 256, 512\}$$

For each window, we train the model to convergence while keeping the architecture fixed.

To separate temporal support from sample support, we use two sampling regimes. In **Exp A**, the sequence is sampled at a base sampling rate f_s and tokenized with window size M_Δ. In **Exp B**, the sampling rate is doubled to $2f_s$, and the window size is doubled to $2M_\Delta$. Thus the temporal support of a token, $\Delta = \frac{M_\Delta}{f_s}$, is held fixed across the two experiments, while the number of samples within the same window is doubled in Exp B. This construction is designed to observe any differences in the estimation term. If performance improves from Exp A to Exp B, the gain can be attributed to increased sample support rather than to a change in temporal scale.

Throughout this appendix, Exp A and Exp B are plotted at matched temporal support. The x-axis reports the base-grid window size M_Δ. For the same plotted value of M_Δ, Exp A uses M_Δ samples at sampling rate f_s, whereas Exp B uses $2M_\Delta$ samples at sampling rate $2f_s$. Thus, the compared points have the same temporal duration Δ, but Exp B has twice the sample support within that duration.

We use three core tasks, plus a burst-noise variant. The details are given below.

Task 1: Frequency-band classification. We generate signals using Equation 13.

$$x[t] = A \sin\left(2\pi f \frac{t}{f_s} + \phi\right) + \varepsilon[t], \qquad \varepsilon[t] \sim \mathcal{N}(0, \sigma^2), \tag{13}$$

where $A > 0$ and $\phi \in [0, 2\pi)$ are random, and the class label $y \in \{1, \ldots, K\}$ is determined by the frequency interval containing f. This task evaluates how tokenization affects the estimation of a structured local statistic. When the SNR is low or the number of classes K is large, the task becomes more estimation-limited.

Task 2: Phase-bin classification. We fix the frequency $f = f_0$ and vary the phase as follows,

$$x[t] = A \sin\left(2\pi f_0 \frac{t}{f_s} + \phi\right) + \varepsilon[t], \tag{14}$$

where the class label is determined by the phase bin,

$$y = \left\lfloor \frac{K\phi}{2\pi} \right\rfloor$$

This task is sensitive to local alignment and therefore evaluates both sides of the tradeoff. Larger windows provide more samples for stable estimation, but excessive aggregation can blur phase-sensitive structure.

Task 3: Waveform-family classification. We fix a base frequency f_0 and classify the waveform family by generating signals as follows,

$$x[t] = A\, s_y\left(2\pi f_0 \frac{t}{f_s} + \phi\right) + \varepsilon[t], \tag{15}$$

where $s_y(\cdot)$ is one of several periodic waveforms (e.g., sinusoid, square, sawtooth, triangle, pulse). Unlike pure frequency estimation, this task depends on local nonsinusoidal shape and higher-order harmonic content. Accordingly, it is more approximation-limited. When tokenization is applied with a longer window, it should compress all within-window variation into a single token without losing waveform-specific structure.

Task 4: Frequency classification with burst noise. To test robustness to non-stationary corruption, we add burst noise to the signal in Equation 13, where the burst noise is defined as in Equation 16.

$$x_{\text{burst}}[t] = x[t] + \sum_{b=1}^{B} \alpha_b \, \psi_b(t - \tau_b), \tag{16}$$

where each ψ_b is a square or triangular pulse, τ_b is a random burst location, and α_b is a random amplitude. This variant tests whether larger windows can improve robustness by averaging over local transients, or whether small windows are preferable when bursts carry task-relevant structure.

For all tasks except **Task 3**, which has five classes, we set the number of classes to $K = 64$. Representative examples of the synthetic tasks are shown in Figure 12.

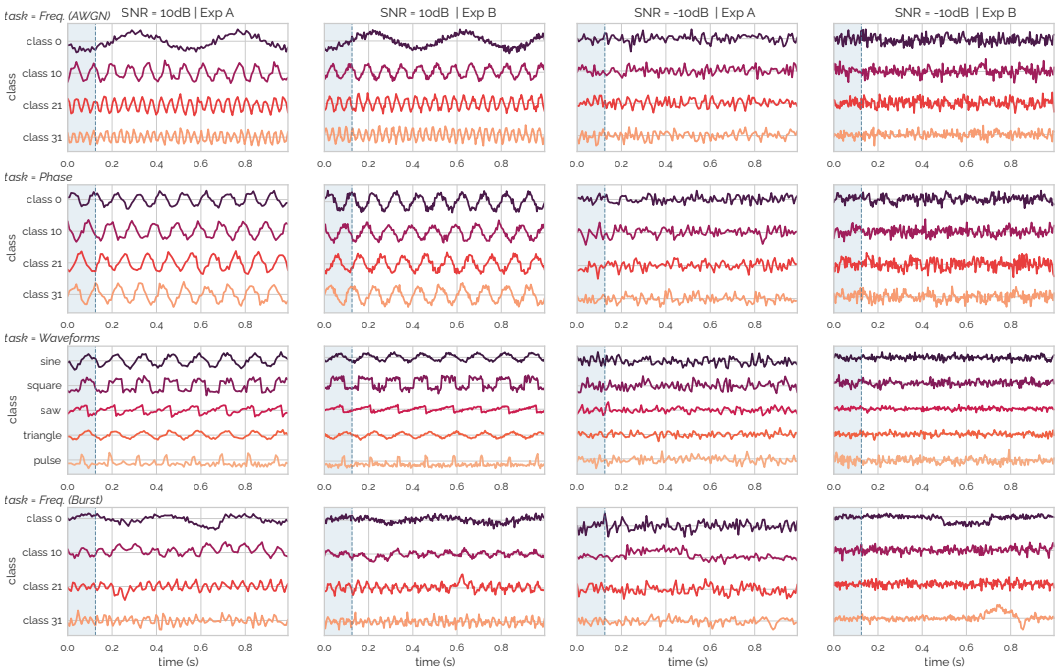

Figure 12: **Synthetic tasks and protocol.** Each row corresponds to one synthetic task. From top to bottom, the rows show frequency-band classification under AWGN, phase-bin classification, waveform-family classification, and frequency-band classification under burst noise. The left and middle columns illustrate Exp A and Exp B, respectively, at 10 dB SNR, where Exp B uses twice the sampling rate and twice the window length so that the temporal support Δ is fixed while the sample support increases. The two rightmost columns show the same comparison at -10 dB SNR.

J.2 Tasks and the Theory

The synthetic tasks are chosen to stress different regimes of the bound

$$\mathbb{E}\left[\|\hat{\theta} - \theta^\star\|^2\right] \lesssim A_{\text{task}} \, \Delta^{2r} + B_{\text{task}} \, \Delta^{-(p-1)} \tag{17}$$

Here A_{task} is a proxy for the approximation and B_{task} is a similar proxy for the estimation.

For frequency-band classification, A_{task} is relatively small because the underlying signal within a clean window is simple, but B_{task} can be large under low SNR. For waveform classification, the opposite tendency holds. The local signal contains sharper structure and more high-frequency content, making A_{task} larger. Phase-bin classification lies between these two extremes, since it requires both stable estimation and local alignment. The burst-noise variant perturbs the estimation term locally and allows us to test the performance of windowing to non-stationary noise.

J.3 Main Effect of Windowing Across Tasks

Figure 13 summarizes test accuracy as a function of windows for the tasks.

First, the optimal length is task-dependent rather than universal. Frequency-band classification prefers intermediate windows over a broad range of conditions. This is consistent with the fact that the task is primarily estimation-limited. However, when the signal is noisy *Blue* lines (-10 dB), the longest window sizes suffer performance degradation.

Second, waveform classification prefers much smaller patches. This task depends on preserving local shape, so its approximation burden grows rapidly with window length. Therefore, the approximation term dominates earlier, causing performance to deteriorate once the window becomes too large.

Third, phase-bin classification exhibits an intermediate regime. Neither the smallest nor the largest windows are uniformly best. Instead, the preferred length shifts with SNR, which is consistent with a mixed regime in which both approximation and estimation contribute.

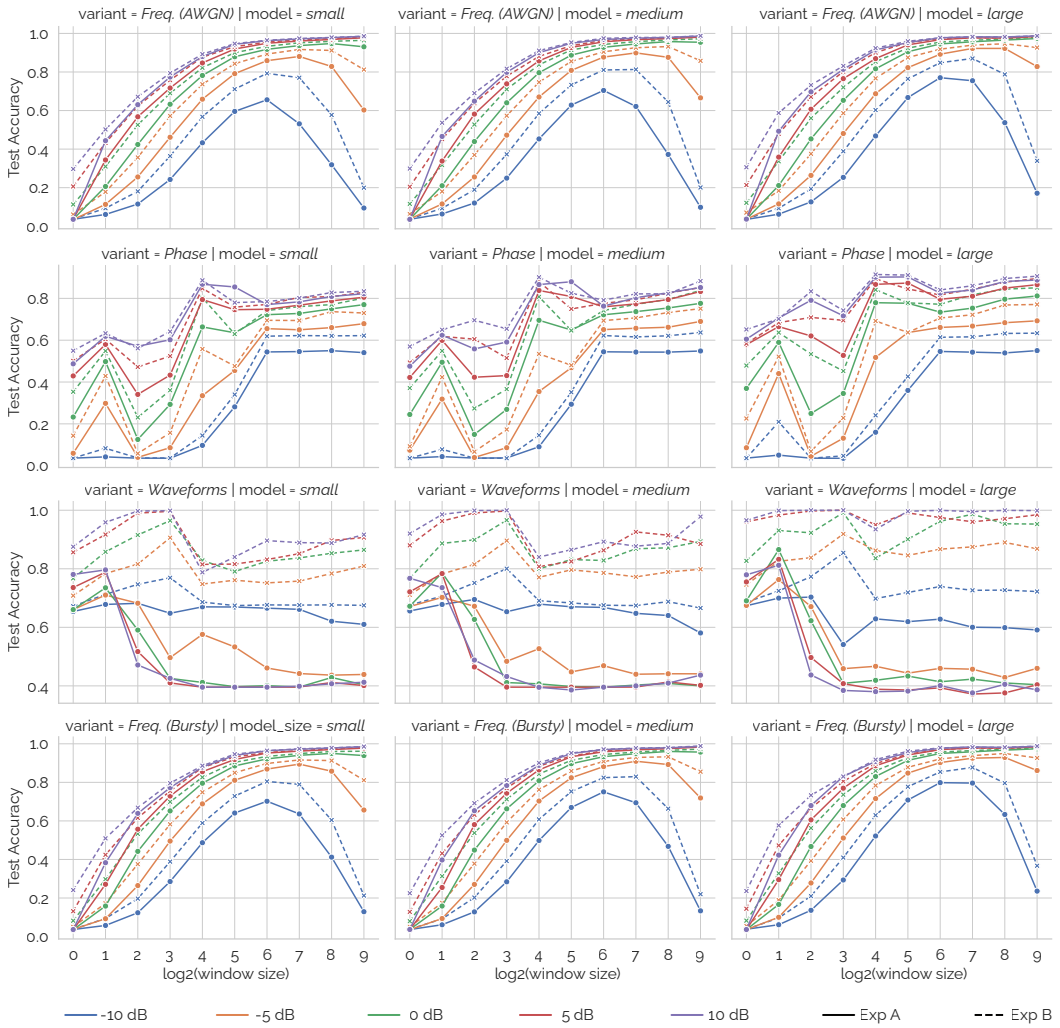

Figure 13: **Window-size sensitivity across tasks.** Test accuracy as a function of $\log_2 M_\Delta$. Columns correspond to different model sizes. At each plotted value of M_Δ, Exp B has the same temporal support as Exp A but twice the sample support. Colors indicate the SNR level, as shown in the legend. The optimal window size depends on both the task and the corruption level.

Overall, the best window size shifts with task type and difficulty, model size, and SNR, as predicted by the theoretical tradeoff. In next sections, we detailed the results.

J.4 Prediction Uncertainty and the Tradeoff

To complement the accuracy curves, Figure 14 reports predictive entropy as a function of window size ($\log_2 M_\Delta$). Predictive entropy serves as an empirical proxy for the model's uncertainty. Analyzing these curves reveals several distinct phenomena that validate the theoretical tradeoff.

The U-Shaped Tradeoff. In the estimation-limited regime (small window sizes), entropy is high because each token contains too few samples to reliably identify the frequency content. As window size grows, entropy drops sharply. However, once the window becomes too large, the approximation burden dominates (the right side of the U), and the model's uncertainty rises again as task-relevant local structure is compressed into a single token representation.

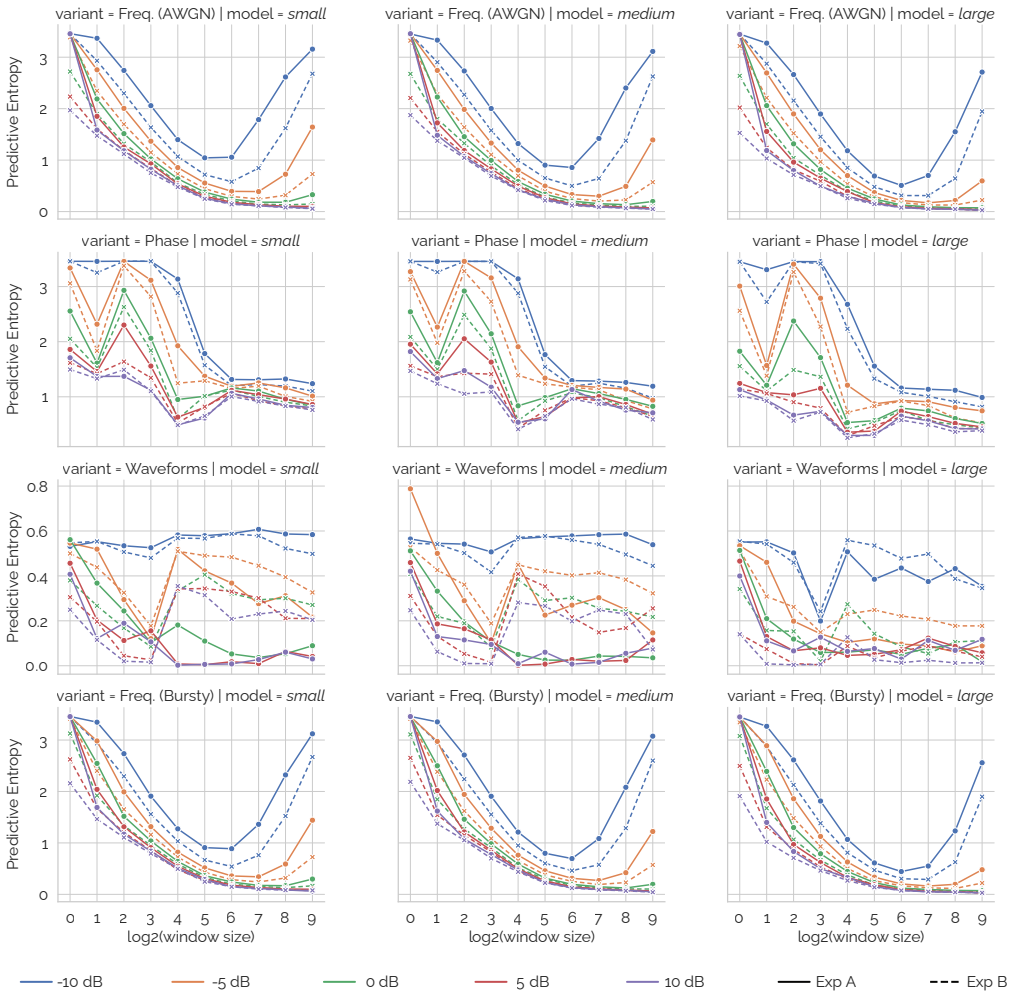

Figure 14: **Predictive entropy versus window length.** Predictive entropy as a function of window length. Lower entropy indicates greater certainty.

The SNR. In the frequency-band settings, the shape of curves depends heavily on the SNR. For low-SNR, the curves exhibit a U-shape. Entropy is high at small windows, drops to a minimum at intermediate, and shoots back up at large window sizes where the combination of noise and compression overwhelms. In high-SNR regimes, the estimation burden is lower; the model achieves high confidence earlier and maintains a stable of low entropy even at the largest window sizes.

Model Capacity. Comparing the *small* and *large* models reveals that increased capacity makes the model more robust to the approximation burden (A_{task}) at larger windows. For example, in the Freq (AWGN) at $-10\,\text{dB}$, the entropy of the *small* model rises to > 3.0 at $\log_2(M_\Delta) = 9$. In contrast,

the *large* model suppresses this rise, keeping entropy below 2.8. This suggests that higher-capacity attention mechanisms might disentangle the information compressed within very large windows.

J.5 Separating Temporal Support from Sample Support

A key advantage of the synthetic setup is that comparing Exp A (base f_s) and Exp B (high f_s) isolates the role of sample density from temporal coverage. Because Exp B doubles both the sampling rate and the token window size, the temporal support Δ remains fixed. Figure 15 plots the accuracy gain ($\text{Acc}_{\text{ExpB}} - \text{Acc}_{\text{ExpA}}$) at matched temporal scales. Positive values indicate regimes where performance is fundamentally limited by sample density rather than temporal context.

Frequency and Burst Tasks (Estimation Dynamics). In the frequency-band tasks, the benefit of increased sample support depends on SNR and window size. Under high SNR, extra samples give an accuracy gain (> 0.15) only at the smallest window sizes. Here, the base model is sample-starved; once the window is large enough, the base sampling rate provides data, and the gain drops to zero. Conversely, under low SNR, the estimation burden is severe. The model requires far more samples to average out the noise, resulting in sustained accuracy gains across most window sizes and a distinct spike at extreme window sizes ($\log_2 = 9$) where the base model completely fails.

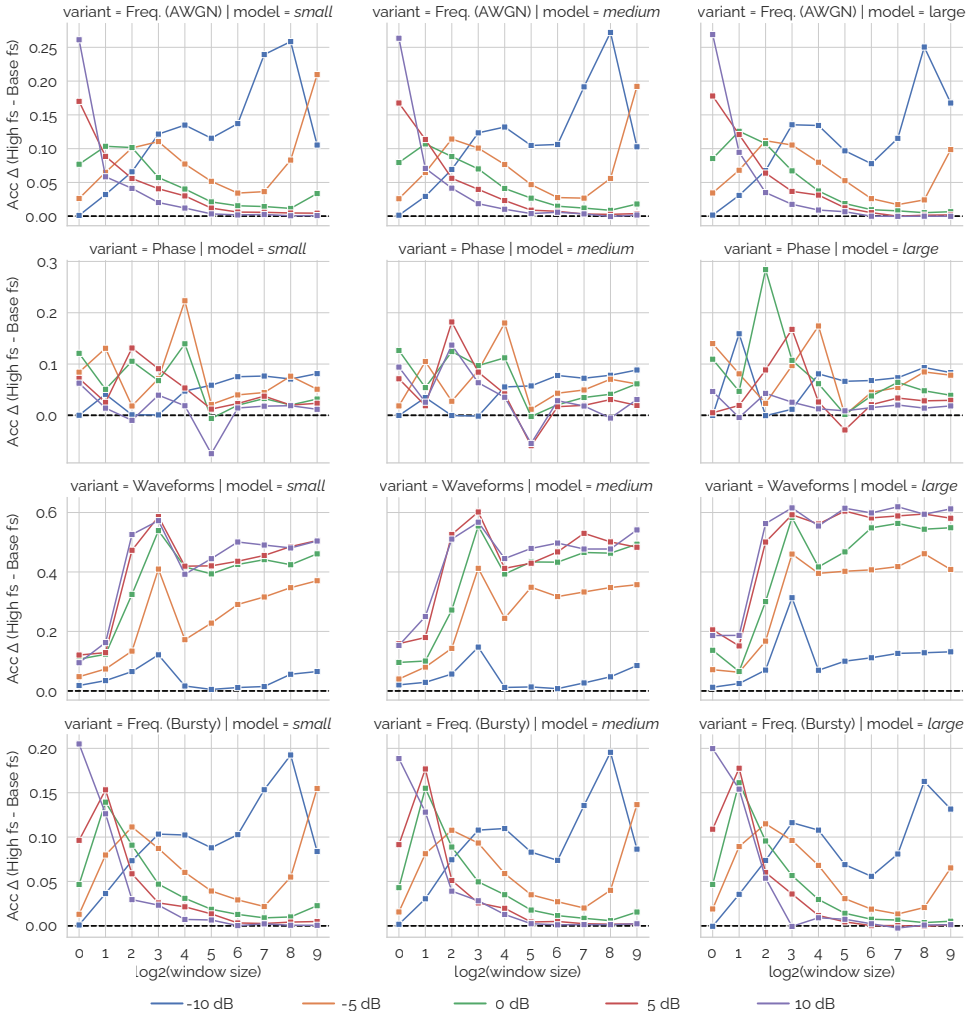

Figure 15: **Benefit of increased sampling resolution at fixed temporal support.** Accuracy gain of Exp B over Exp A as a function of window size with different noise levels. Positive values indicate that additional samples within the same temporal window improve estimation.

Waveform Task. Waveform classification exhibits massive gains from increased sampling rates, reaching up to 60% absolute improvement. Waveform classification depends on distinguishing sharp, high-frequency shapes (e.g., the edges of square or sawtooth waves). Thus, extra samples reduce the approximation burden by increasing window resolution.

Phase Task. The gains in this tasks are volatile across window sizes. While higher sampling rates provide improvements in resolving phase, the curves reflect that this task is sensitive to where the boundaries align with the signal's cycle where increasing sample density cannot entirely resolve.

J.6 Differentiating Low and High Frequencies

The frequency discrimination task also allows a finer question. It answers whether the larger windows necessarily suppress the ability to distinguish high-frequency components. Our experiments show that the different frequency bands can be extracted independent of window size.

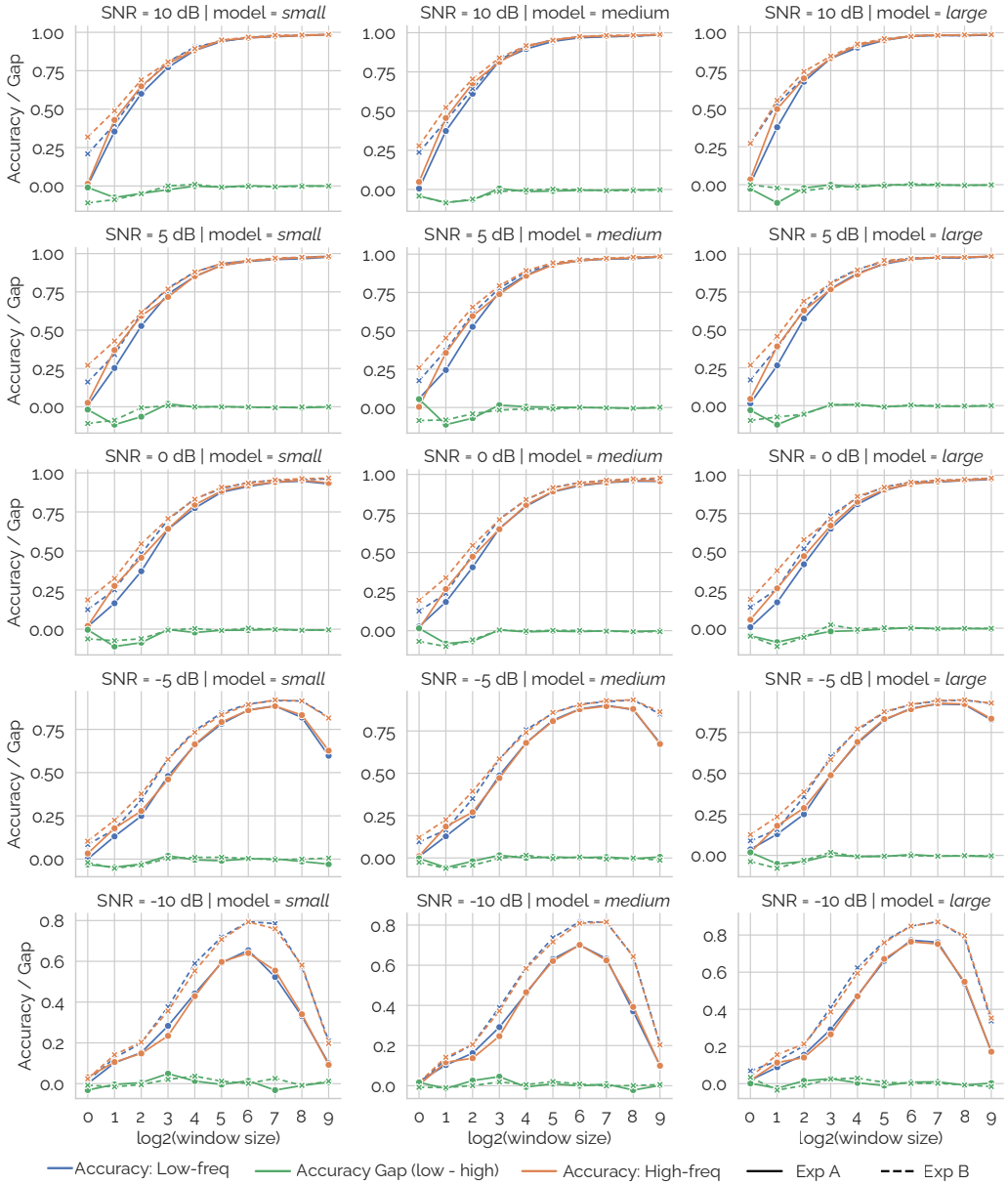

Figure 16: **Low versus high-frequency discrimination across window sizes.** Accuracy on low and high-frequency class groups, together with their gap, as a function of windows.

Recent work suggests that models exhibit an implicit frequency bias, in the sense that large windows may behave like low-pass filters that favor low-frequency structure over high-frequency variation [57, 73]. Our results suggest that this is an incomplete heuristic. As shown in Figure 16, the accuracy gap between low and high-frequency groups remains close to zero even for large windows.

Rather than acting as an inherent low-pass filter, windowing changes the approximation-estimation tradeoff. The effects noted in prior work are consistent with an improved estimation term, since larger windows average over noise, whereas degradation on more within-window variation is consistent with the approximation term, since a longer window must compress more variation into a single token. When high-frequency structure is task-relevant and the model has sufficient representational capacity, large windows can retain it as effectively as low-frequency structure.

Figure 16 supports this interpretation. The accuracy gap remains near zero for almost all window sizes, across SNR levels and model sizes. At extreme window sizes under high noise, both frequency groups degrade similarly. At very small window sizes ($\log_2 M_\Delta \leq 2$), the gap is occasionally negative which indicates that high-frequency classes can be slightly easier to classify in this regime.

These results show that larger windows do not selectively discard high-frequency. Instead, failures at large window sizes arise because the approximation burden increases when too much variation must be compressed into a single token. The observed changes in accuracy are therefore better explained by the developed tradeoff in our theory than by a frequency-specific suppression mechanism.

We also conducted an additional control experiment in which each high-frequency signal was augmented with a low-frequency component as in Equation 18.

$$x[t] = A_h \sin\left(2\pi f_h \frac{t}{f_s} + \phi_h\right) + A_\ell \sin\left(2\pi f_\ell \frac{t}{f_s} + \phi_\ell\right) + \varepsilon[t], \qquad f_\ell \ll f_h, \tag{18}$$

while the class label was still determined only by the high-frequency component. Empirically, this modification produced almost no change in classification performance. This suggests that, even when low and high-frequency information coexist in the same window, gradient-based learning can still recover the task-relevant component rather than collapsing to the lower-frequency trend.

Mathematically, a signal can be represented a superposition of frequency components, and the downstream objective depends only on the statistic associated with f_h. Thus, provided the tokenizer and encoder have sufficient capacity, the model can learn a representation that is approximately invariant to the nuisance low-frequency term while preserving the discriminative high-frequency structure. This again indicates that failures at large window sizes are better explained by the general approximation–estimation tradeoff than by an unavoidable suppression of high-frequency information.

J.7 Non-Stationary Burst Noise

The burst-noise condition provides a controlled test of robustness to non-stationary corruption. Figure 17 plots the accuracy difference ($\Delta_{\text{acc}} = \text{Acc}_{\text{bursty}} - \text{Acc}_{\text{AWGN}}$) to isolate the specific impact of localized transients across window sizes.

Vulnerability at Small Windows. At shorter windows ($\log_2 \leq 2$), the accuracy gap is negative across all noise levels. Because shorter windows lack temporal context, a localized burst can dominate the token's statistics, destroying the underlying representation. In this regime, the model suffers a severe estimation failure. At the same time, the results are not monotone. When the window becomes too large, the benefit of averaging can be offset by excessive aggregation of unrelated structure.

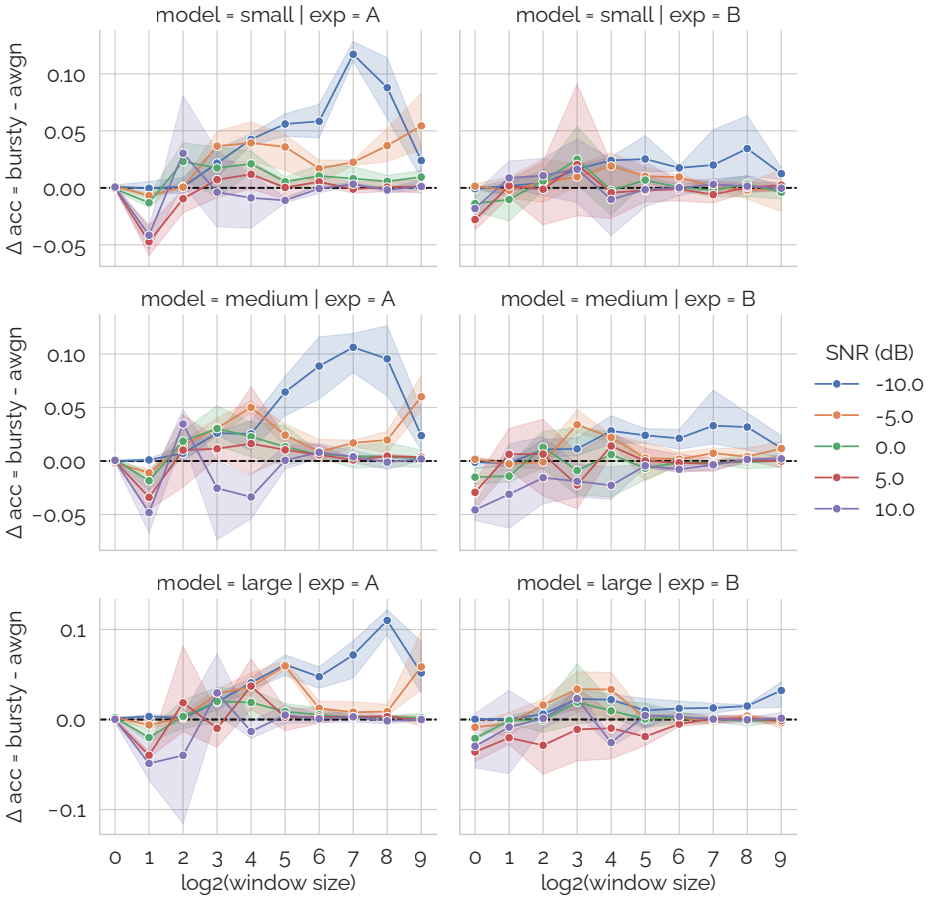

Figure 17: **Window-size sensitivity under non-stationary burst noise.** Accuracy difference between the burst-noise and AWGN conditions as a function of $\log_2 P$, shown across SNR levels and model sizes. Positive values indicate a benefit under burst corruption relative to the stationary-noise.

Averaging at Larger Windows. As the window size increases, the model gains context to average over the localized bursts. In high-SNR regimes, the performance gap stabilizes near zero for $\log_2 \geq 3$. However, in low-SNR, the gap becomes positive at large window sizes ($\log_2 \in [6, 8]$). This indicates that background noise (AWGN) worsen the approximation burden of tokens more severely than burst spikes do, because the large window can still capture clean signal segments between the bursts.

Mitigation via Sample Density (Exp A vs. Exp B). In Exp B, the performance degradations and fluctuations seen in Exp A are suppressed. Doubling the sampling rate within the same temporal window provides sufficient density for the model to separate the high-frequency burst transients from the underlying task-relevant signal.

J.8 Dimension of the Space

To determine whether downstream representational capacity can compensate for sub-optimal tokenization, Figure 18 plots accuracy against the token embedding dimension (`dim`). We compare the performance of the best and worst window sizes for each configuration across both SNR regimes.

Window Length Dominates Embedding Dimension. One important observation is the gap between the optimal (blue) and worst (orange) window sizes. The accuracy of the worst window size remains flat across most embedding dimensions and models. This demonstrates a fundamental information bottleneck. If a window is poorly chosen to capture statistics or too large to compress task-relevant information. Increasing the dimensionality of the space cannot recover information that was lost.

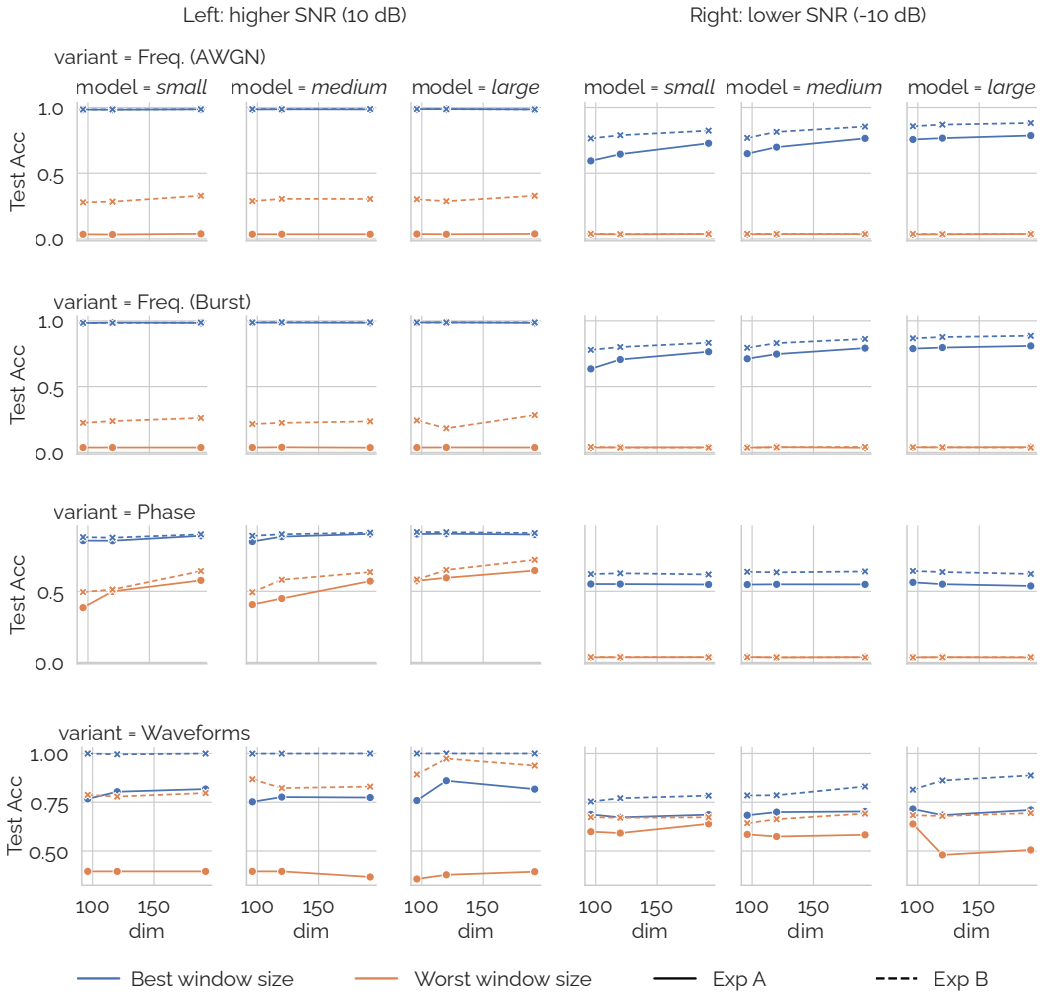

Figure 18: **Effect of embedding dimension.** Accuracy is shown as a function of embedding dimension for the best and worst window sizes across tasks, models, and two SNRs. Across settings, increasing the dimension does not close the gap caused by a sub-optimal window size, showing that representational capacity cannot fully compensate for information lost during tokenization.

Marginal Gains at Optimal Lengths. For the optimal windows, increasing the embedding dimension provides only marginal benefits. Under low-SNR conditions, there is a slight, gradual upward trend in accuracy as `dim` increases (e.g., in Freq AWGN and Freq Burst). This suggests that a wider embedding space can help the attention mechanism in filtering out heavy noise, but only if the window size is already good.

Sample Density vs. Latent Capacity. The plot reinforces the superiority of Exp B (dashed lines) over Exp A (solid lines). In almost all scenarios where performance is not already saturated, doubling

the raw sample density within the token (Exp B) yields a larger accuracy boost than doubling the latent embedding dimension. This confirms that representation learning is primarily bounded by raw signal resolution and estimation support, rather than the width of the latent vector space.

J.9 Training and Scaling

One remaining question is whether the observed behavior is consistent with the dataset size.

Dataset. For every synthetic task variant and condition, we use 20,000 training samples, 4,000 validation samples, and 4,000 test samples. We performed additional scaling experiments by increasing the training set size by up to $8\times$. As expected, performance improves when the task is not already saturated. Importantly, the same qualitative phenomena remain after scaling, namely that the preferred window size continues to depend on the task and noise. This suggests that the observed trends are not only a small-data artifact, but persist even as the amount of training data increases.

Optimization. All models are trained for 10 epochs We use a learning rate of 3×10^{-4}, a weight decay of 0.01, and a dropout of 0.1. The batch size is set to 2048 for the *small* and *medium* architectures. For the *large* model, the batch size is reduced to 1024.

Reproducibility. A global random seed is set for the initialization of all environments. The dataset generation is entirely deterministic, utilizing offset seeds to guarantee distinct, non-overlapping distributions between the training, validation, and test splits while being consistent across experiments. For distributed training, all Distributed Data Parallel (DDP) ranks instantiate identical dataset tensors, which are deterministically partitioned across GPUs using seeded distributed samplers.

J.10 Connection Back to the Theory

The synthetic experiments can be interpreted directly through the bound

$$\mathbb{E}\left[\|\hat{\theta}-\theta^\star\|^2\right] \le L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r} + \frac{a}{T}\Delta^{-(p-1)}, \qquad a = c f_s^{-p} \tag{19}$$

The first term measures how difficult it is to compress a window into a token, through the parameters C_{app} and r, while the second measures how difficult it is to estimate the window statistic from finitely many samples, through the estimation exponent p and the variance constant a.

The synthetic tasks cover different regimes of the bound. For frequency-band classification, the signal within a window is relatively simple, so the approximation burden is mild, corresponding to a smaller effective C_{app}. At the same time, the task is estimation-limited, especially at low SNR, so the variance term is dominant for small windows. This is why larger windows often help until the approximation term becomes non-negligible. For waveform-family classification, the signal contains sharper local structure and richer harmonic content, making it harder to approximate by a token. This corresponds to a larger effective C_{app} and a less favorable approximation regime, so the bias term dominates earlier and the preferred window size shifts smaller. Phase-bin classification lies between these two regimes, since it requires both stable estimation and preservation of local alignment.

The Exp A/B comparison probes the effect of increased sample support at fixed temporal duration. Across these experiments, the temporal support $\Delta = \frac{M_\Delta}{f_s}$ is held fixed, while the sampling rate is doubled in Exp B. Therefore, the approximation term $L_\phi^2 C_{\mathrm{app}}^2 \Delta^{2r}$ remains unchanged. By contrast, the variance constant scales to

$$a_{\mathrm{B}} = c(2f_s)^{-p} = 2^{-p} a_{\mathrm{A}}$$

Because the estimation term scales with $\Delta^{-(p-1)}$, it blows up at small windows. The proportional reduction by 2^{-p} in Exp B explains why we observe the most significant accuracy gains at the small window sizes ($\log_2 \le 3$) for the frequency and phase tasks (Figure 15).

The burst-noise is also consistent with the theorem. Burst noise increases the local estimation difficulty, which can be interpreted as increasing the constant c and therefore increasing a. This amplifies the variance term for small windows. As a result, larger windows can become more favorable because they reduce the sensitivity to localized corruption, until the approximation term again becomes dominant.

Overall, the synthetic results support the interpretation of the theorem. The experiments therefore do not only show that window size matters; they show that the observed optima move in the direction predicted by the approximation–estimation tradeoff developed in the theory.

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: The abstract and introduction state the claims made, including the contributions.

   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 paper has a separate Limitation section (See Section 8).

   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:[Yes]

Justification: We have a comprehensive theoretical result section (See Appendix A) including assumptions.

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 have a dedicated implementation section in Appendix D.

   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 have provided anonymized scripts to reproduce the results.

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 paper has a specific section (See Appendix D) for the implementation details.

   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 have reported the error bars, obtained with different random seeds, for all the experiments we performed.

   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: The details are given in Appendix Section D.

   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: Authors have conducted the research according to the NeurIPS 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: [N/A]

    Justification: The concept of temporal sequence tokenization is broad and not tied to a specific application. Since the proposed method is also generic, we have not included potential positive or negative impacts.

    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: Our analysis and method do not pose a risk.

    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 cite the original sources for all existing assets used in the paper. The assets are publicly available, and we have checked that their stated terms of use permit research use.

    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: [Yes]

    Justification: The only new asset, the code, introduced in the paper is well documented with license, etc.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: The paper does not involve crowdsourcing nor research with human subjects.

    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: [N/A]

    Justification: LLMs are only used for paraphrasing and grammar check.

    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.