TSketches: A New Primitive for Efficient Time Series Transformers

Rundong Zuo
Hong Kong Baptist University
csrdzuo@comp.hkbu.edu.hk

Rui Cao
Hong Kong Baptist University
csrcao@comp.hkbu.edu.hk

Guozhong Li
KAUST
guozhong.li@kaust.edu.sa

Byron Choi
Hong Kong Baptist University
bchoi@comp.hkbu.edu.hk

Jianliang Xu
Hong Kong Baptist University
xujl@comp.hkbu.edu.hk

Sourav S Bhowmick
Nanyang Technological University
assourav@ntu.edu.sg

Abstract

Transformer models have recently been applied to time series and exhibited great performance. However, we identify a key limitation of existing time series transformers is the absence of an effective tokenization mechanism for extracting semantically meaningful tokens. In particular, most time series transformers directly take every individual raw value as a token (a.k.a value- based tokens), which ignores rich semantics present in time series subsequences. Furthermore, taking numerous value tokens as inputs suffers from the well-known quadratic training time complexity of transformers. To address this, we propose a *new time series primitive*, called TSketches, which serves as semantic tokens for the transformers. We propose a novel framework, called TSTUDIO, that comprises two main components to extract TSketches: (1) a stratified representation learning framework that combines iSAX-LSH with a novel STRATified-negAtives triplet loss (STRATALOSS) to learn prototypes of TSketches, serving as a vocabulary of a time series dataset; and (2) a TSKETCHER that takes time series as input and searches the nearest subsequences to prototypes as TSketches. Extensive experiments on eight longest UCR/UEA datasets, synthetic datasets, and five transformers verify the effectiveness of TSketches on the downstream classification task, where TSketches lead to 2–120× training speed-ups while consistently achieving higher accuracy.

PVLDB Reference Format:
Rundong Zuo, Rui Cao, Guozhong Li, Byron Choi, Jianliang Xu, and Sourav S Bhowmick. TSketches: A New Primitive for Efficient Time Series Transformers. PVLDB, 17(11): XXX-XXX, 2024.
doi:XX.XX/XXX.XX

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/rdzuo/tsketches.

1 Introduction

Time series data is extensively generated across diverse domains such as healthcare, smart cities, finance, and economics [3, 19, 28].

Proceedings of the VLDB Endowment, Vol. 17, No. 11 ISSN 2150-8097.
doi:XX.XX/XXX.XX

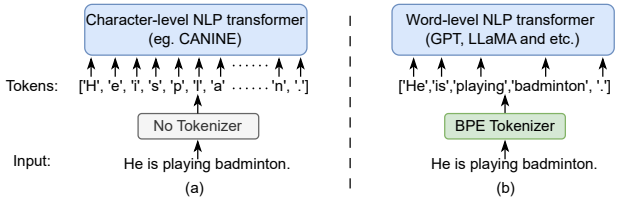


Figure 1: (a) NLP transformer without tokenizer. (b) NLP transformer with tokenizer, *e.g.*, GPT, LLaMA, and others [4, 10, 31, 35].

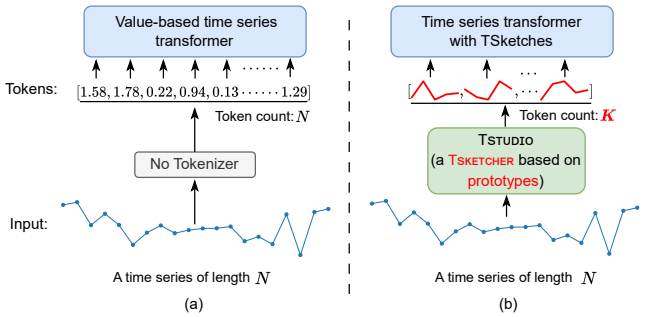


Figure 2: (a) Value-based time series transformer, *e.g.*, [44]. (b) Time series transformer with TSketches obtained from TSTUDIO.

Despite significant advances, time series analysis remains a central research focus in both academia and industry [17, 18, 33, 48]. In parallel, transformer-based models have achieved success in NLP by effectively modeling long-range dependencies through self-attention mechanisms [12, 30, 38].

It is evident that there exists a natural tokenization of text, which decomposes text into discrete tokens and maps them to dense embeddings. Tokens are typically words/subwords, as they carry richer semantics than individual characters. For example, character-level models (*e.g.*, CANINE [9]) process raw Unicode of characters as input (Figure 1(a)), while word-level models employ a BPE tokenizer [5, 36] (Figure 1(b)). This approach not only reduces the input length but also provides richer semantic information.

In parallel to the success of transformers, researchers have applied them to time series with promising results [7, 21, 44, 50]. However, few of them have explored the semantic tokens for time series transformers, leaving several challenges unresolved.

Challenges: We identify two key challenges in existing time series transformers. ① Most models, referred to as *value-based transformers* [44, 46, 47], directly input individual raw time series values into the transformer (Figure 2(a)). Since transformers suffer from the well-known quadratic computational complexity of the attention

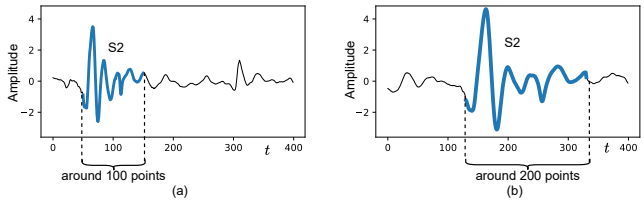


Figure 3: The *globe scaling* of time series. S2 is the *second heart sound* pattern of *heartbeat sound*. Due to variations in heart rate, S2 (blue) of the two time series differ in their temporal or amplitude scales.

mechanism [34, 38], representing each time point as a token makes training inefficient and memory-intensive for long sequences. Furthermore, unlike word tokens in NLP, raw time series values are low-level and lack semantic meaning, which limits their representational power. ② Recent approaches, termed *shape-based transformers* [21, 49, 50], build tokens from subsequences or shapelets. Although such tokens capture part of the semantics, they still remain two limitations: (i) the discovered shapes often capture only local patterns [23] and are not designed to form semantic vocabulary across datasets, and (ii) shape discovery is computationally expensive [22].

Our approach: We remark that the main reason for the above challenges is the absence of an effective tokenization mechanism for long time series. To fill in the gap, we propose TSketches, a new time series primitive serving as semantic tokens for transformer input (Figure 2(b)), extracted from raw series by our Tstudio. Specifically, Tstudio first learns a set of prototypes to form a reusable time series vocabulary. Then, for a given input series, it efficiently searches the approximate best-matching subsequence of each prototype as TSketch. Thus, a long time series of length N is represented by K semantic tokens, namely TSketches, with $K < N$, reducing sequence length while improving efficiency and scalability.

The details of Tstudio are illustrated in Figure 4. In the offline phase, Tstudio learns a set of semantically meaningful prototypes from subsequence collections. We first leverage iSAX-LSH to map subsequences into discrete buckets, and define the iSAX bucket distance with a probabilistic order-preserving (POP) guarantee. This ensures that bucket distances reflect semantic similarity, enabling stratification across multiple levels. On top of this stratification, we propose a STRATified-negAtives triplet loss (Strataloss), which assigns level-specific margins instead of a fixed one as in prior work [15, 22]. To further handle *global scaling* [41], where semantically similar shapes occur at different temporal or amplitude scales (shown in Figure 3), we randomly apply a kernel from a predefined set $\mathcal{H}$ during training. Together, these mechanisms yield discriminative, semantically rich subsequence representations whose centers form prototypes, serving as a vocabulary for time series.

Next, for each prototype, we identify its approximate best-matching subsequence in the input time series, called a TSketch. One may apply a sliding window to obtain subsequences, but searching across all subsequences for every prototype is computationally expensive ($O(NK)$). Moreover, conventional similarity search methods [6, 14, 24, 39] assume pre-indexed subsequences for ad-hoc queries, which are inapplicable here since subsequences are generated on-the-fly from each input series. Therefore, we propose TSketcher, an index built on prototypes, which efficiently searches TSketches with complexity reduced to $O(N \log K)$. Each subsequence is routed to the candidate set of its nearest prototype, ensuring that the final match for each prototype not only preserves efficiency but also retains semantic consistency. As summarized in Table 1, TSketch tokens not only shorten training time, but also preserve richer semantics and improve search efficiency compared to existing value- or shape-based tokens.

Table 1: Comparison of training and search complexity across value-based, shape-based, and sketch-based transformers.

Property	Value-based	Shape-based	TSketches
Tokens' semantic richness	Low	Variable	High
Training complexity	$O(N^2)$	$O(K^2)$	$O(K^2)$
Search complexity	N/A	$O(NK)$	$O(N \log K)$

We conduct extensive experiments on the eight longest time series datasets from the UCR/UEA archives [11], synthetic datasets of varying scales, and five transformer backbones. Empirical results demonstrate that using TSketches as transformers' inputs accelerates the training time of a vanilla transformer by 7-120× and other state-of-the-art efficient transformers by 2-40×. In terms of accuracy, TSketch-based models achieve superior or highly competitive performance on most datasets. Our main contributions are summarized as follows:

- We propose a new time series primitive, called TSketches, which are subsequences as semantic tokens. We propose Tstudio, a framework that transforms long time series into TSketches. By using K TSketches instead of N values from a long time series to train transformer models, the training time complexity is reduced from $O(N^2)$ to $O(K^2)$.
- We propose a stratified representation learning through a novel Strataloss. The Strataloss utilizes the iSAX data structure as LSH to stratify negatives to multiple levels, and handles the *global scaling* by applying a kernel to anchor, thus learning a set of semantically-rich prototypes of TSketches as vocabulary for time series.
- We propose TSketcher, which efficiently searches the approximate matching subsequence for each prototype as a TSketch from an input time series. The time series is then represented by K TSketches. The search time complexity is $O(N \log K)$.
- We conduct extensive experiments on real and synthetic datasets, demonstrating the effectiveness of TSketch as a new primitive. They achieve clear speedups and accuracy gains. Furthermore, we present a case study using the BinaryHeartbeat dataset.

Organization. The rest of this paper is organized as follows. Some background knowledge is presented in Section 2. Section 3 presents the details of Tstudio. Section 4 reports the experimental results. Section 5 reviews the related works. Section 6 concludes the paper and presents our future work.

2 Background

In this section, we introduce the key terminologies and notations. Table 2 summarizes some frequently used symbols.

Time series instance. A time series instance (or simply a *time series*) is denoted as $X = (x_1, x_2, \cdots, x_N)$, where N is the number

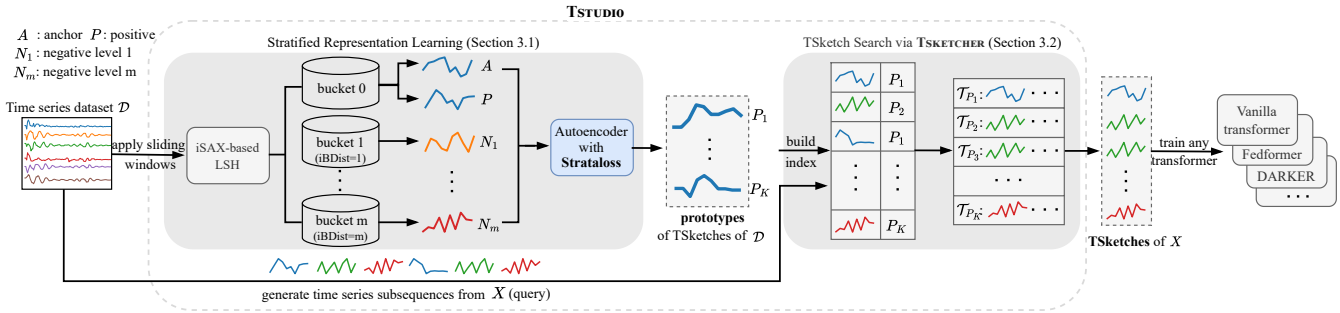


Figure 4: The overview of our TSTUDIO.

of timestamps/observations. A time series dataset $\mathcal{D}$ contains a collection of time series instances $|\mathcal{D}|$.

Time Series Subsequences. Given a time series $X = (x_1, x_2, \cdots, x_N)$, a subsequence $T[l:r]$ is defined as the contiguous segment $(x_l, x_{l+1}, \cdots, x_r)$, where $1 \le l \le r \le N$. Here, l and r denote the start and end timestamps of the subsequence, respectively.

2.1 Self-attention mechanism of transformers

The core part of the transformer model is the self-attention mechanism, which captures long-range dependencies by modeling pairwise interactions among all input tokens. This process can be described as a function that maps a query and a set of key-value pairs to an output. The self-attention mechanism in the vanilla transformer is formalized as:

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \tag{1}$$

where Q, K, and $V \in \mathbb{R}^{N\times d}$ represent the query, key, and value matrices, respectively, for a sequence of N input tokens of dimension d. This formulation computes pairwise similarity scores between all queries and keys, applies softmax normalization, and uses the resulting attention weights to combine the value vectors. While this mechanism enables the model to capture dependencies across arbitrary distances in the sequence, it requires $O(N^2)$ time complexity, making it the major computational bottleneck [13, 34].

We now describe the derivation of the Q, K, and V matrices from input tokens. Given an input sequence $X = \{x_1, x_2, \ldots, x_N\} \in \mathbb{R}^{N\times d}$, where each $x_i \in \mathbb{R}^d$ is a input token. Then:

$$Q = XW_q, \quad K = XW_k, \quad V = XW_v, \tag{2}$$

where W_q, W_k, and $W_v \in \mathbb{R}^{d\times d}$ are learnable weights. For the existing value-based transformer (Figure 1), each data point is treated as a token, so a series of length N produces N tokens. Since transformer complexity scales quadratically with the number of tokens, training becomes prohibitive for long sequences.

2.2 Triplet Loss in Time Series

Triplet loss [15] is a recent contrastive objective, defined over triplets of an anchor a, a positive p (similar to a), and a negative n (dissimilar to a). The loss enforces a margin constraint to separate positives from negatives:

$$D_{AP} + \alpha < D_{AN} \tag{3}$$

where $D(\cdot,\cdot)$ is a distance metric (*e.g.*, Euclidean distance) and $\alpha > 0$ is a predefined margin. Minimizing this loss encourages the model to learn an embedding space where similar samples form compact clusters, while dissimilar ones are separated.

Table 2: Frequently Used Notations

Notation	Description
X	a time series instance, consists of N time points
T	a time series subsequence generated from X
S_X	TSketches of a time series X
K	the number of TSketch s in S_X
$\mathcal{P}$	the set of prototypes
P	a prototype in $\mathcal{P}$
H	a CNN kernel with fixed weights
$\mathcal{H}$	the set of kernels

ShapeNet [22] extends the triplet framework with a cluster-wise triplet loss, where positives and negatives are defined by k-means clustering, namely positives (the same cluster) and negatives (different clusters). It further introduces an intra-cluster regularization term D_{neg} to constrain negative samples:

$$D_{neg} = \max_{i,j\in(1,K^-)\wedge i<j}\left\{\|f(x_i) - f(x_j)\|_2^2\right\} \tag{4}$$

where x denotes negative samples and K^- is the number of negatives. This term minimizes the maximum pairwise distance among the negatives, encouraging them to stay close together.

These approaches still suffer from a key limitation. Equation 3 applies a fixed margin to all negatives, while Equation 4 penalizes large pairwise distances among negatives. As a result, negatives that are originally far apart in the raw space may collapse into nearby regions in the embedding space in an epoch of the training. The fundamental issue is that they treat all negatives indiscriminately, ignoring their inherent dissimilarity structure. In contrast, our proposed STRATALOSS explicitly stratifies negatives into multiple levels, allowing level-specific margins to preserve these differences.

Problem statement. Given a time series $X = (x_1, x_2, \ldots, x_N)$, the goal is to extract a set of representative subsequences $S_X = \{s_1, s_2, \ldots, s_K\}$, termed TSketches, which serve as input tokens to a transformer model. □

3 Time Series Studio (TSTUDIO)

In this section, we propose TSTUDIO to extract time series TSketches as input tokens for transformers. As shown in Figure 4, the framework consists of two main modules: (1) a stratified representation learning to obtain prototypes of TSketches via STRATALOSS; (2) a TSKETCHER for efficiently searching the approximate best-matching TSketch for each prototype.

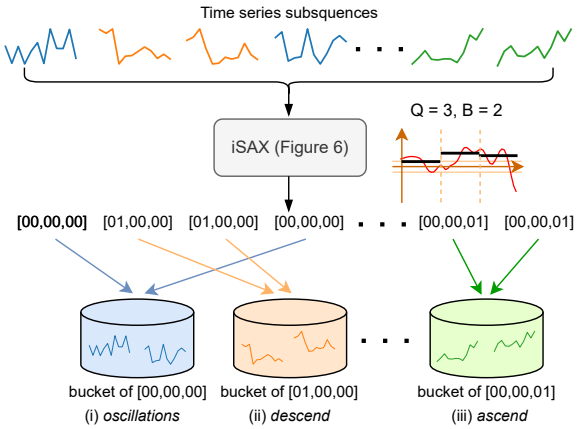


Figure 5: Illustration of iSAX-based Locality-Sensitive Hashing (iSAX-LSH). Similar subsequences (in the same color) are hashed to one bucket, yet share the same semantics signified by their shapes (*e.g.*, subsequences in blue are (i) *"oscillations"*, orange are (ii) *"descend"*, and green are (iii) *"ascend"*).

3.1 Stratified Representation Learning

To obtain semantically rich prototypes of TSketches, we propose a stratified representation learning framework consisting of two steps: (a) stratification via iSAX-LSH and (b) train an autoencoder with StrataLoss.

3.1.1 Stratification via iSAX-LSH A central challenge in contrastive learning in time series is how to stratify negative samples for an anchor subsequence. Unlike text, which naturally provides discrete semantic units (*e.g.*, words), subsequences are unlabeled numerical fragments, making it difficult to assign them to meaningful similarity levels.

To address this, we propose a stratification process via iSAX-LSH to capture the semantics with iSAX bucket distance, which has the Probabilistic Order-Preserving (POP) property in Theorem 3.1. In particular, we adopt iSAX [6, 32], a symbolic approximation that segments a subsequence into Q parts and quantizes each part into a B-bit code. This yields an initial symbolic word that captures the semantics of the subsequence signified by its shape. We then apply locality-sensitive hashing (LSH) on these codes to group subsequences into buckets: those mapped to the same bucket are likely to share similar semantics, while the distance between buckets provides a principled way to define multiple levels of semantic dissimilarity.

Formally, given a time series subsequence $T_i = \{x_l, x_{l+1}, \ldots, x_r\}$, the iSAX hash value is defined as:

$$(b_i^1, b_i^2, \ldots, b_i^Q) = h_{\text{iSAX}}(T_i), \tag{5}$$

where Q is the number of equal-length segments, and $b_i^Q \in \{0,1\}^B$ is the B-bit representation for each segment.

The resulting hash values have a length of $Q * B$ bits, and subsequences sharing the same hash value are assigned to the same bucket. Figure 5 illustrates the process: each subsequence is mapped to an iSAX hash value, and subsequences with identical iSAX hash values are co-located in the same bucket, implying similar semantic information. For example, we observe that the subsequences in blue, orange, and green are*oscillation*, *descend*, and *ascend*, respectively.

Based on the iSAX hash value, we further define a distance metric, *iSAX bucket distance*, to measure the distance between buckets, as presented in Definition 1:

Definition 1. ***(iSAX bucket distance)*** *Given two time series subsequences $T = (t_1, \ldots, t_n), U = (u_1, \ldots, u_n)$, and their corresponding iSAX hash values $b_T = (b_T^1, \ldots, b_T^w), b_U = (b_U^1, \ldots, b_U^w)$, where each segment b_T^k, b_U^k is a B-bit binary code, the distance between the k-th segments is defined as follows:*

$$\text{dist}_{iB}(b_T^k, b_U^k) = |\textstyle\sum_{i=0}^{B-1} 2^i \cdot (a_i - c_i)|, \tag{6}$$

where $b_T^k = a_{B-1} \ldots a_0$ and $b_U^k = c_{B-1} \ldots c_0$.

The iSAX bucket distance between b_T and b_U is then defined as:

$$\text{iBDist}(b_T, b_U) \equiv \textstyle\sum_{k=1}^{w} dist(b_T^k, b_U^k) \tag{7}$$

It is worth noting that, unlike traditional iSAX-based measures such as MinDist [25], which are designed to lower-bound the Euclidean distance and thus preserve absolute distances, our iSAX bucket distance possesses a key property called *Probabilistic Order-Preserving (POP)* focusing on relative similarity ordering among subsequences. Specifically, given two pairs of subsequences, if $iBDist(b_{T_i}, b_{T_j}) < iBDist(b_{T_k}, b_{T_l})$, then with high probability (w.h.p.) their true Euclidean distances satisfy $Dist(T_i, T_j) < Dist(T_k, T_l)$. This property enables us to stratify negatives into multiple levels, where the distance between buckets reflects the increase in dissimilarity, laying the foundation for learning discriminative and semantically rich prototypes. We then formally state and prove the POP property in Theorem 3.1.

Theorem 3.1 (**Probabilistic Order-Preserving Property of iSAX bucket distance**). *Let $\mathcal{N}(\boldsymbol{\mu}, \Sigma)$ denote the normal distribution with mean $\boldsymbol{\mu}$ and variance Σ. Let w be the number of PAA segments, and let $T = (t_1, \ldots, t_n), U = (u_1, \ldots, u_n)$ be two time series subsequences with $t_1, \ldots, t_n, u_1, \ldots, u_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. Assume their iSAX bucket distance is $iBDist(b_T, b_U) = d_{iB}$. Then, for any $\epsilon, \delta > 0$, the following inequality holds:*

$$\begin{aligned}\Pr\Big[\|T-U\|_2^2 \le n\delta^2 + d_{iB}\tfrac{n}{w}(d_z^2 + 2\delta d_z) + 2(n-w) + \epsilon\Big] \\ \ge \big(2\Phi(\sqrt{\tfrac{n}{2w}}(d_z+\delta)) - 1\big)^{d_{iB}} \cdot \big(2\Phi(\sqrt{\tfrac{n}{2w}}\max(\delta, d_z)) - 1\big)^{w-d_{iB}} \\ \cdot \frac{\epsilon^2}{8n(1-\frac{w}{n})^2 + \epsilon^2},\end{aligned} \tag{8}$$

where $d_z = \Phi^{-1}(1 - \frac{1}{2^B}) - \Phi^{-1}(1 - \frac{2}{2^B})$ is a constant, $\Phi(\cdot)$ is CDF of the standard Gaussian distribution and B is the number of iSAX bits per segment.

Proof.

$$\|T-U\|_2^2 = \tfrac{n}{w}\textstyle\sum_{i=1}^{w}(\bar{t}_i - \bar{u}_i)^2 + \sum_{i=1}^{w}\sum_{j=1}^{\frac{n}{w}}[(t_{ij} - \bar{t}_i) - (u_{ij} - \bar{u}_i)]^2 = A + G \tag{9}$$

Let $\{z_i\}_{i=1}^{2^B-1}$ be the quantile points in each segment. According to the property of iSAX, $d_z = \Phi^{-1}(1 - \frac{1}{2^B}) - \Phi^{-1}(1 - \frac{2}{2^B})$ where $\Phi(\cdot)$ is the CDF of the Gaussian distribution. We have the inequality below:

$$\Pr[|\bar{t}_i - \bar{u}_i| \le d_z + \delta] = 2\Phi(\sqrt{\tfrac{n}{2w}}(d_z + \delta)) - 1 \tag{10}$$

There exists:

$$\begin{aligned}\Pr[A \le n \cdot \max(\delta, d_z)^2 + d_{iB} \cdot \frac{n}{w}(\min(\delta, d_z)^2 + 2\delta d_z)] \\ \ge (2\Phi(\sqrt{\frac{n}{2w}}(d_z+\delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w-d_{iB}}\end{aligned} \tag{11}$$

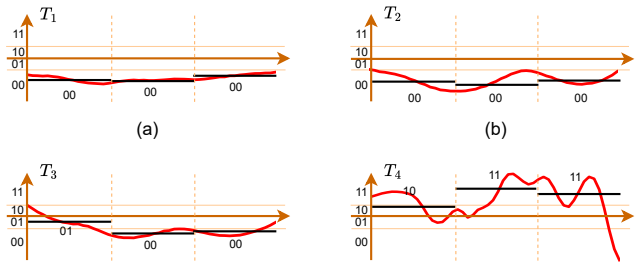


Figure 6: Examples of iSAX values.

Then, we analyze the distribution of G. We obtain $\mathbb{E}[G] = 2(1 - \frac{w}{n}) \cdot n = 2(n - w)$ and $\text{Var}(G) = 8n(1 - \frac{w}{n})^2$. According to the one-sided Chebyshev inequality, for $\forall \epsilon > 0$, the following holds

$$\Pr[G - 2(n - w) \geq \epsilon] \leq \frac{8n(1-\frac{w}{n})^2}{8n(1-\frac{w}{n})^2+\epsilon^2}. \tag{12}$$

After combining Equations 11 and 12, we have:

$$\begin{aligned}&\Pr[\|T - U\|_2^2 = A + G \leq D_A(d_{iB}, \delta) + D_G(\epsilon)] \\ &\geq (2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w-d_{iB}} \\ &\cdot \frac{\epsilon^2}{8n(1-\frac{w}{n})^2 + \epsilon^2}.\end{aligned} \tag{13}$$

□

Here, we summarize key equations to illustrate the intuitions behind our iSAX bucket distance, and the detailed proof is provided in the Appendix A.1. Intuitively, the probabilistic bound in Equation 13 is mainly determined by the term $(2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w-d_{iB}}$, while $\frac{\epsilon^2}{8n(1-\frac{w}{n})^2+\epsilon^2}$ can be close to one since ϵ is an arbitrary positive value. Thus, the final probability is dominated by d_{iB} and δ in the former term since n, w and d_z are given. Furthermore, a smaller d_{iB} leads to a higher probability of yielding a tighter upper bound on the Euclidean distance between T and U. Thus, subsequences with similar iSAX hash values can achieve a high probability of lying within a small-radius sphere in the original space.

By contrast, the MinDist metric [25] in traditional iSAX is designed as a lower bound to facilitate fast retrieval [6], but it offers weaker probabilistic guarantees of relative similarity order under normal distribution assumptions on input time series subsequences. A key limitation lies in its treatment of adjacent intervals: MinDist assigns a distance of zero when two PAA segments fall into neighboring intervals, whereas our iSAX bucket distance assigns a value of one. Consequently, MinDist cannot distinguish subsequences in adjacent symbolic intervals from those with identical symbols.

To further clarify the computation of iSAX bucket distance and its POP property in Theorem 3.1, we provide a concrete example with four subsequences in Example 1 compared to MinDist.

Example 1. *Assume that we have four time series subsequences T_1, T_2, T_3, and T_4, each of length 50, as shown in Figure 6. Their iSAX values are $[00, 00, 00]$, $[00, 00, 00]$, $[01, 00, 00]$, and $[10, 11, 11]$, respectively. Thus, T_1 and T_2 are assigned to the same bucket $[00, 00, 00]$, T_3 to $[01, 00, 00]$, and T_4 to $[10, 11, 11]$.*

The Euclidean distance, traditional iSAX distance (MinDist), and iSAX bucket distance between T_1 and the others are as follows:

Pair	*Euclidean*	*MinDist*	*iSAX bucket dist.*
(T_1, T_2)	*2.06*	*0*	*0*
(T_1, T_3)	*4.03*	*0*	*1*
(T_1, T_4)	*12.94*	*8.21*	*8*

It should be noted that MinDist is 0 to the distance between the first segments of T_1 and T_3 though they are in neighboring intervals with iSAX values 00 and 01 in Figure 6(a) and 6(c), respectively.

Example 1 provides three key insights. First, T_1 and T_2, the pair with the smallest Euclidean distance, are correctly hashed to the same bucket, showing that iSAX-LSH effectively groups similar subsequences. Second, the order of iSAX bucket distances, $dist_{iB}(T_1, T_4) > dist_{iB}(T_1, T_3)$, preserves the ordering of their Euclidean distances, $dist(T_1, T_4) > dist(T_1, T_3)$. This empirically validates the POP property in Theorem 3.1, confirming that the iSAX bucket distance can serve as a reliable proxy for the semantic token space for distance-based sampling and stratification. Third, our iSAX bucket distance can distinguish T_1 and T_3, whereas MinDist treats them as the same ones, again indicating the limitation of MinDist on the POP property.

3.1.2 *Training autoencoder with* StrataLoss

With negatives stratified into multiple levels by iSAX-LSH, the next step is to learn embeddings that preserve semantic information in a low-dimensional space. To achieve this, we adopt an autoencoder $model_A$ consisting of an encoder $f_e(\cdot)$ and a decoder $f_d(\cdot)$ as model structure and propose a novel StrataLoss for training.

Assigning multiple margins based on iSAX bucket distance. StrataLoss leverages bucket-based stratification to assign leveled margins between positives and negatives. Specifically, in each training batch, an anchor A and a positive P are sampled from the same iSAX-LSH bucket, while all remaining subsequences from other buckets serve as negative candidates. These negatives are stratified into multiple levels based on their iSAX bucket distances from the anchor's bucket.

Figure 7 contrasts our method with previous triplet-based approaches [15, 22, 42]. At the top of the figure, the samples are organized into a positive set and multiple negative levels based on iSAX bucket distances (different shapes indicate different levels). As shown in Figure 7(b), previous methods enforce a fixed margin (Equation 3), which collapses negatives from distinct semantic levels and obscures their true separations. In contrast, StrataLoss assigns level-specific margins that increase with the iSAX distance in Figure 7(c), with the standard triplet loss as a special case when m=1.

Formally, for a negative sample N_i at level i, we enforce the following constraint in the embedding space:

$$D_{AP} + \frac{i}{m}\alpha < D_{AN_i}, \tag{14}$$

where D_{AP} and D_{AN_i} denote the distances between the anchor–positive and anchor–negative pairs, respectively, m is the total number of negative levels (with $m = Q * (2^B - 1)$), and α is the maximum margin parameter.

This formulation ensures that positives remain closer to the anchor than negatives, with the margin increasing with the negative level i. As a result, semantically similar subsequences are mapped

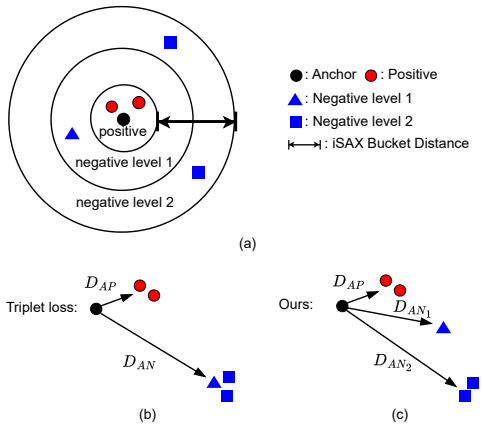


Figure 7: (a) The negative levels determined by iSAX bucket distances. (b) Existing triplet loss, setting a fixed margin for all negatives. (c) StrataLoss, setting multiple margins based on the (a).

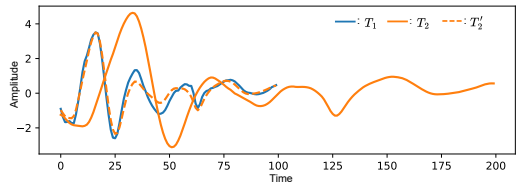


Figure 8: Apply kernel techniques to tackle *global scaling* of T_1 and T_2.

closer together in the embedding space. Conversely, dissimilar subsequences are pushed farther apart, such that the separation distance reflects their iSAX bucket distance. As a result, by setting multi-level margins to prevent samples from collapsing into a single negative class, StrataLoss yields a low-dimensional embedding space that is not only highly discriminative, but also preserves the semantic iSAX structure, thereby facilitating prototype extraction.

Based on Equation 14, for a negative sample x_{n_i} (a subsequence) at level i, the corresponding term in loss function is computed by:

$$\mathcal{L}_i = \max\left(d(f_e(x_a), f_e(x_p)) - d(f_e(x_a), f_e(x_{n_i})) + \frac{i}{m}\alpha, 0\right) \quad (15)$$

where $d(\cdot,\cdot)$ denotes the distance, α is the maximum margin, and $f_e(\cdot)$ is the encoder.

Applying kernel techniques for *global scaling*. As we have shown in Figure 3, subsequences that differ in temporal or amplitude scale may have the same semantic meaning (*i.e.*, the same prototype), which is called *global scaling* in time series. To mitigate the impact of global scaling issues on prototypes, we incorporate a scale-consistency term $\mathcal{L}_{sc}$:

$$\mathcal{L}_{sc} = d(f_e(x_a), f_e(x_a')) \quad (16)$$

where x_a' is obtained by applying a convolution kernel H to the anchor x_a:

$$x_a' = H * x_a \quad (17)$$

The kernel H is selected from a kernel set $\mathcal{H}$ that contains kernels with various strides. The weights for these kernels are sampled from a Gaussian distribution centered on a mean filter.

Figure 8 provides an example of how the use of convolutional kernels addresses the *global scaling*. T_1 and T_2 are the two subsequences mentioned in Figure 3. After convolving the longer subsequence T_2 with a kernel whose weights are $[0.38, 0.38]$, and a stride of 2, we obtain T_2', which is now much closer in distance to T_1. This reduced

Algorithm 1: Training $model_A$ with StrataLoss

Input: Time series subsequences $\mathcal{T}$ obtained from $\mathcal{D}$, the iSAX buckets $\mathcal{B}$ of each $\mathcal{T}$, the kernel set $\mathcal{H}$
Output: The trained model $model_A$

1 $model_A$.init(); // Initialize $model_A$ consists of a encoder $f_e(\cdot)$ and a decoder $f_d(\cdot)$
2 **for** $epoch \in \{0, \cdots, epochs\}$ **do**
3 // Computing StrataLoss
4 $x_a, x_p, b_a \leftarrow$ Random select($\mathcal{T}, \mathcal{B}$);
5 $\{x_{n_1}, x_{n_2}, x_{n_3}, \cdots, x_{n_m}\} \leftarrow$ SelectNegative($\mathcal{T}, \mathcal{B}, b_a$);
6 **for** i *in* $range(1, m+1)$ **do**
7 $\mathcal{L}_i \leftarrow \max\left(d(f_e(x_a), f_e(x_p)) - d(f_e(x_a), f_e(x_{n_i})) + \frac{i}{m}\alpha, 0\right)$ // Equation 15
8 $\mathcal{L}_n$ += $\mathcal{L}_i$;
9 $H \leftarrow$ Random pick($\mathcal{H}$);
10 $\mathcal{L}_{sc} \leftarrow d(f_e(x_a), f_e(H * x_a))$ // Equation 16
11 $\mathcal{L}_{Strataloss} \leftarrow \mathcal{L}_{sc} + \frac{1}{m}\mathcal{L}_n$ // Equation 18
12 // Computing reconstruction loss
13 **for** T_i *in* $\mathcal{T}$ **do**
14 $\mathcal{L}_{r_i} = \|T_i - f_d(f_e(T_i))\|^2$ // Equation 20
15 $\mathcal{L}_{re}$ += $\mathcal{L}_{r_i}$;
16 $\mathcal{L}_{all} \leftarrow \lambda\mathcal{L}_{triplet} + (1-\lambda)\mathcal{L}_{re}$ // Equation 19
17 $model_A$.backward($\mathcal{L}_{all}$);
18 **return** $model_A$

distance ensures that the model perceives them as semantically similar, thereby treating them as TSketches of the same prototype during training. $\mathcal{L}_{sc}$ minimizes their embedding distance, thereby enhancing prototype invariance to scale variations.

The final Stratified-Negatives triplet loss is defined as the sum of $\mathcal{L}_{sc}$ and the average triplet loss across all negative levels $\mathcal{L}_i$:

$$\mathcal{L}_{Strataloss} = \mathcal{L}_{sc} + \frac{1}{m}\sum_{i=1}^{m}\mathcal{L}_i \quad (18)$$

Meanwhile, since we adopt an autoencoder structure, to ensure reconstruction fidelity in the learned embedding space, we formulate a composite loss function $\mathcal{L}_{all}$ that linearly combines our proposed Stratified-Negatives triplet loss $\mathcal{L}_{Strataloss}$ and a reconstruction loss $\mathcal{L}_{re}$:

$$\mathcal{L}_{all} = \lambda\mathcal{L}_{Strataloss} + (1-\lambda)\mathcal{L}_{re} \quad (19)$$

where $\lambda \in [0, 1]$ is a hyperparameter to balance their contributions. $\mathcal{L}_{re}$ is defined as the mean squared error (MSE) between x_i and its reconstruction $\hat{x}_i$ via the autoencoder:

$$\mathcal{L}_{re} = \|x_i - f_d(f_e(x_i))\|^2 \quad (20)$$

where $f_e(\cdot)$ and $f_d(\cdot)$ denote the encoder and decoder, respectively.

Algorithm 1 presents the details of stratified representation learning. The input consists of time series subsequences $\mathcal{T}$ generated by sliding windows on $\mathcal{D}$, along with the corresponding iSAX bucket for each $T \in \mathcal{T}$. For clarity, we omit the standard mini-batch partitioning and present a simplified training view over the full training set across all epochs. We begin by initializing the autoencoder $model_A$, consisting of an encoder $f_e(\cdot)$ and a decoder $f_d(\cdot)$ (Line 1). For each epoch (Line 2), we randomly sample an anchor x_a, and positives x_p (Line 4). Negative samples are stratified into m levels based on their iSAX bucket distance to b_a (Line 5). The stratified triplet loss $\mathcal{L}_n$ is then computed by aggregating the losses across all negative levels (Lines 6–8). Meanwhile, we compute $\mathcal{L}_{sc}$ by randomly

Algorithm 2: Prototype Extraction

Input: Time series subsequences $\mathcal{T}$ obtained from $\mathcal{D}$, the trained model $model_A$
Output: prototypes $\mathcal{P}$

1 Initialize set $\mathcal{R}$, $\mathcal{P}$;
2 // Computing the representations $\mathcal{R}$ based on $f_e(\cdot)$
3 **for** T *in* $\mathcal{T}$ **do**
4 $R \leftarrow f_e(T)$;
5 $\mathcal{R}$.append(R);
6 $C : \{C_1, C_2, ..., C_K\} \leftarrow$ kmeans($\mathcal{R}$);
7 // Computing the prototypes $\mathcal{P}$ based on $f_d(\cdot)$
8 **for** C *in* C **do**
9 $P \leftarrow f_d(C)$;
10 $\mathcal{P}$.append(P);
11 **return** $\mathcal{P}$

picking a kernel H from the kernel set $\mathcal{H}$ (Lines 9-10). The overall StrataLoss is then calculated as $\mathcal{L}_{sc} + \frac{1}{m}\mathcal{L}_n$ (Line 11). Subsequently, the reconstruction loss $\mathcal{L}_{re}$ is computed across all subsequences $T_i \in \mathcal{T}$ via the autoencoder (Lines 12–15). Finally, $\mathcal{L}_{all}$ is formed as the weighted sum of StrataLoss and $\mathcal{L}_{re}$ (Line 16), and used to update the model parameters via backpropagation (Line 17).

Prototype Extraction. After training the autoencoder $model_A$ detailed in Algorithm 1, we extract representative prototypes of time series subsequences, as summarized in Algorithm 2. First, the encoder $f_e(\cdot)$ projects all subsequences $\mathcal{T}$ into low-dimensional representation space, generating the set $\mathcal{R}$ (Lines 2–5). Next, K-Means clustering is applied to $\mathcal{R}$ to obtain K cluster centers $C = \{C_1, C_2, \ldots, C_K\}$ (Line 6), which are the latent representations of prototypes. Each cluster center is then decoded back into the original time series space via $f_d(\cdot)$, producing the final prototype set $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$ in the original time series space (Lines 7–10).

3.2 TSketch Search via TSketcher

In Section 3.1.2, we have constructed a set of K prototypes $\mathcal{P}$ for a time series dataset. To efficiently compute the tokens for transformers from a long time series, we formulate the problem of TSketch search as follows:

[**TSketch search problem**] Given a prototype set $\mathcal{P} = \{P_1, P_2, \ldots, P_K\}$, for an input time series $X = \{x_1, x_2, \ldots, x_N\}$, the *TSketch search problem* is to find a set of TSketches $S_X = \{s_1, s_2, \ldots, s_K\}$ for X, such that

$$s_i = \arg\min_{T_j \in \mathcal{T}} dist(T_j, P_i), \tag{21}$$

where $\mathcal{T}$ is the set of subsequences by applying a sliding window to X.

The primary challenge of the TSketch search problem lies in achieving this with sub-linear complexity relative to the $O(NK)$ brute-force approach in previous works [21, 49, 50]. In addition, the search problem cannot be directly addressed by conventional similarity search techniques [6, 14, 24, 39, 41], as they typically build indexes on a large number of subsequences for ad-hoc queries. In contrast, in our problem, the subsequences are generated from the input time series X, which serve as the *queries*, and are determined to be TSketches when they best-match the prototypes (Equation 21). Such a query paradigm is analogous to publish/subscribe systems,

Algorithm 3: TSketches Searching from Prototypes $\mathcal{P}$

Input: a time series X of length N, prototypes $\mathcal{P}$, a *BallTREE* built on $\mathcal{P}$, $\mathcal{P}_{nearest}$ the nearest neighbors of $\mathcal{P}$
Output: TSketch set S of time series X

1 Initialize TSketch set $S = \emptyset$ for X;
2 Initialize $\{\mathcal{T}_{P_k} = \emptyset\}_{k=1}^{K}$;
3 $\mathcal{T} : \{T_1, T_2, T_3, ..., T_N\} \leftarrow$ slideTS(X);
4 // Search the nearest prototypes to map $\mathcal{T} = \{T_1, T_2, T_3, ..., T_N\}$
5 **for** T *in* $\{T_1, T_2, T_3, ..., T_N\}$ **do**
6 $P_k \leftarrow BallTREE$.query(T);
7 $\mathcal{T}_{P_k}$.append(T)
8 // Search TSketch for each prototype P_i in its candidate set $\mathcal{T}_{P_i}$
9 **for** P_i *in* $\mathcal{P}$ **do**
10 // If P_i is not assigned to any T as its label, take $\mathcal{T}_{P'}$ as $\mathcal{T}_{P_i}$
11 **while** $\mathcal{T}_{P_i} == \emptyset$ **do**
12 // Take the nearest leaf node P' to P
13 $P' \leftarrow \mathcal{P}_{nearest}$.get($P_i$);
14 $\mathcal{T}_{P_i} \leftarrow \mathcal{T}_{P'}$;
15 // Search the nearest subsequence in $\mathcal{T}_{P_i}$ to be a TSketch s_i corresponding to prototype P_i
16 $s_i \leftarrow argmin_{T \in \mathcal{T}_{P_i}}(dist(P_i, T))$;
17 S.append(s_i)
18 **return** S

where the subsequences are (i) matched against a relatively small number of prototypes, and (ii) routed to the candidate sets of their prototypes for further matching.

Steps of TSketcher**.** To solve the TSketch search problem, we propose TSketcher, which efficiently computes TSketches that are approximate best-matches of the prototypes with a probabilistic guarantee on recall. The overview of TSketcher is shown in Figure 9. We construct a tree-based index on the prototypes offline once. For a concrete presentation, we simply adopt the ball tree (denoted as BallTree). The steps of TSketcher are also shown in Algorithm 3. In query time, we take an input time series X, apply a sliding window to obtain a set of time series subsequences $\mathcal{T}$: $\{T_1, \ldots, T_N\}$ (Line 3). ① Routing: We query each subsequence T in the index of prototypes for its nearest prototype P_k, and route it to the candidate set $\mathcal{T}_{P_k}$ (Lines 5–7). For example, both blue subsequences have P_1 as their nearest neighbor and they are routed to the same set $\mathcal{T}_{P_1}$.

② Populating: Next, there can be the case that the candidate set $\mathcal{T}_{P_i}$ of a P_i is empty. In this case, we fetch P_i's nearest prototype P' that has non-empty $\mathcal{P}_{nearest}$ and set its candidate set to $\mathcal{T}_{P_i}$ (Lines 10–14). For instance, if any prototype's set remains empty ($\mathcal{T}_{P_3}$ in Figure 9), we populate the empty set with the subsequences from the nearest non-empty set ($\mathcal{T}_{P_2}$ in Figure 9).

③ Searching: Finally, for each prototype P_i, we compute the subsequence in $\mathcal{T}_{P_i}$ that is nearest to P_i as TSketch s_i (Lines 15–17). A baseline method to compute TSketch s_i for a given P_i is to search within $\mathcal{T}_{P_i}$ and its nearby candidate set, which ensures the exact best matching. However, we prove searching within $\mathcal{T}_{P_i}$ alone can guarantee a lower bound on recall of the best matching, as presented in Theorem 3.2. Hence, in Line 16, we search s_i in P_i. In

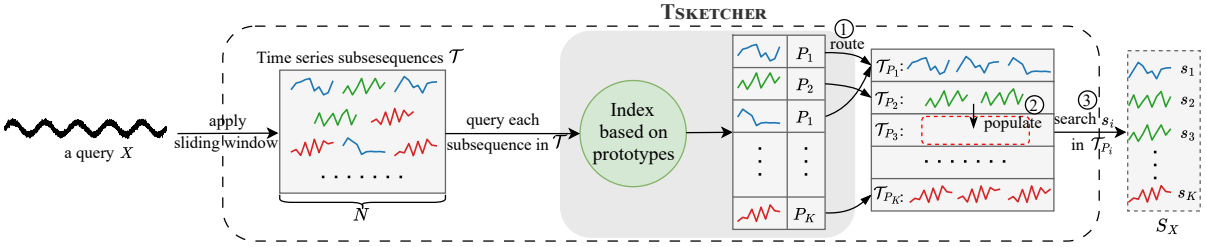


Figure 9: TSketch search process. Subsequences are first mapped to their nearest prototypes with an index. Empty sets are then populated with subsequences the same as the nearest non-empty candidate set. Finally, TSketches are selected as the closest subsequences to each prototype.

Line 18, Algorithm 3 yields a total of K TSketches, denoted as $S_X = \{s_1, s_2, \cdots, s_K\}$. Each TSketch s_i is for the corresponding prototype P_i, which represents X for model training.

Example 2. *Consider two prototypes $P_1 = (0.12, 0.27)$ and $P_2 = (0.89, 0.77)$, along with two subsequences $T_1 = (0.83, 0.72)$ and $T_2 = (0.91, 0.84)$.*
① Routing. Both T_1 and T_2 are closer to P_2 than to P_1, since $dist(P_2, T_1) = 0.08 < 0.84 = dist(P_1, T_1)$ and $dist(P_2, T_2) = 0.07 < 0.97 = dist(P_1, T_2)$. Thus, $\mathcal{T}_{P_1} = \emptyset$ and $\mathcal{T}_{P_2} = \{T_1, T_2\}$.
② Populating. Since $\mathcal{T}_{P_1}$ is empty, $\mathcal{T}_{P_1}$ is populated with subsequences from $\mathcal{T}_{P_2}$, as P_2 is the nearest prototype to P_1. Hence, $\mathcal{T}_{P_1} = \{T_1, T_2\}$.
③ Searching. For P_1, the nearest subsequence in $\mathcal{T}_{P_1}$ is T_1 ($dist(P_1, T_1) = 0.84 < 0.97$). For P_2, the nearest subsequence is T_2 ($dist(P_2, T_2) = 0.07 < 0.08$). The selected TSketches are $s_1 = T_1$ and $s_2 = T_2$.

The probabilistic lower bound of searching the true nearest neighbor of the prototype by using TSketcher on recall is given in Theorem 3.2.

Theorem 3.2 (**Lower bound guarantee of Algorithm 3**). *Given time series prototypes $\mathcal{P} = \{P_1, \ldots, P_K\}$ and time series subsequences $\mathcal{T} = \{T_1, \ldots, T_N\}$ where $P_1, \ldots, P_K, T_1, \ldots, T_N \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d)$ where I_d denotes the $d \times d$ identity matrix, TSketch s_i returned by Algorithm 3 has the following property:*

$$\Pr\left[s_i = \text{NN}_{\mathcal{T}}(P_i)\right] \geq 1 - \exp\left(-\alpha(d) \cdot \frac{N}{K}\right), \tag{22}$$

where $\text{NN}_A(x)$ denotes the nearest neighbor search (NN) of x in set A and $\alpha(d)$ is a positive constant dependent only on the dimension d.

Proof. According to Algorithm 3, we can rewrite the probability with the expectation below:

$$\Pr[s_i = \text{NN}_{\mathcal{T}}(P_i)] = \mathbb{E}_{\mathcal{T},\mathcal{P}}\left[\frac{1}{K}\sum_{i=1}^{K} \mathbf{1}\left(\text{NN}_{\mathcal{P}}\left(\text{NN}_{\mathcal{T}}(P_i)\right) = P_i\right)\right] \tag{23}$$

where $\mathbf{1}$ denote the indicator function. Given $\mathcal{P} = \{P_1, \ldots, P_K\} \subset \mathbb{R}^d$, the Voronoi region of P_i is

$$V(P_i) = \{x \in \mathbb{R}^d \,|\, \|x - P_i\| \leq \|x - P_j\|, \forall j \neq i\} \tag{24}$$

For each P_i, we explore its nearest neighbor $T^* = \text{NN}_{\mathcal{T}}(P_i)$. If $T^* \in V(P_i)$, we have $p_i = \text{NN}_{\mathcal{T}}(P_i)$. Thus the probability holds

$$\Pr[s_i = \text{NN}_{\mathcal{T}}(P_i)] = \mathbb{E}_{\mathcal{T},\mathcal{P}}\left[\frac{1}{K}\sum_{i=1}^{K} \mathbf{1}\left(T^* \in V(P_i)\right)\right] \tag{25}$$

Assume $R_i = \min_{j \neq i} \frac{1}{2}\|P_i - P_j\|$ representing the minimum radius of the Voronoi region of P_i. If $T^* \in B(P_i, R_i)$, then T^* lies in $V(P_i)$. Thus, we calculate the probability $\Pr[\|T^* - P_i\| < R_i] = \Pr[T^* \in V(P_i)]$.

Let $Z_j = \|T_j - P_i\|$ and $z_j = T_j - P_i \sim \mathcal{N}(0, 2I_d)$. Then we achieve $\|z_j\|^2 \sim 2\chi_d^2$ and $Z_j = \|T_j - P_i\| \sim \sqrt{2}\chi_d$. The distribution of the nearest neighbor distance R_{NN} is shown below:

$$F_{NN}(r) = \Pr[\min_j Z_j < r] = 1 - (1 - F_{\chi_d}(\tfrac{r}{\sqrt{2}}))^N \tag{26}$$

Therefore,

$$\Pr[T^* \in V(P_i)] \geq \Pr[R_{NN} < R_i] = 1 - (1 - F_{\chi_d}(\tfrac{R_i}{\sqrt{2}}))^N \tag{27}$$

For $P_i \sim \mathcal{N}(0, I_d)$, we can obtain $\mathbb{E}[R_i] = \mathbb{E}[\frac{1}{2}\min_{j \neq i}\|P_i - P_j\|] = (\frac{2}{K})^{1/d} - (\frac{1}{K})^{1/d} = c_d \cdot K^{-1/d}$ according to the theory on nearest neighbor in [1] where c_d is a constant dependent on d. When $R_i \ll 1$, the Cumulative Distribution Function (CDF) of the χ distribution can be denoted as:

$$\begin{aligned} F_{\chi_d}(\frac{R_i}{\sqrt{2}}) &= \frac{1}{\Gamma(d/2)}\int_0^{R_i^2/4} t^{d/2-1}e^{-t}dt \geq \frac{1}{\Gamma(d/2)}\int_0^{R_i^2/4} t^{d/2-1}dt \\ &= \frac{1}{\Gamma(d/2)} \cdot \frac{(R_i^2/4)^{d/2}}{d/2} = \frac{1}{2^d\Gamma(d/2+1)}R_i^d \end{aligned} \tag{28}$$

Finally, we achieve

$$\begin{aligned} \Pr[R_{NN} < R_i] &\geq 1 - (1 - \frac{1}{2^d\Gamma(d/2+1)}(c_d \cdot K^{-1/d})^d)^N \\ &= 1 - (1 - \alpha(d)K^{-1})^N \geq 1 - \exp(-\alpha(d) \cdot \frac{N}{K}) \end{aligned} \tag{29}$$

and $\alpha(d) = \frac{c_d^d}{2^d\Gamma(d/2+1)}$, *i.e.*, $\Pr[s_i = \text{NN}_{\mathcal{T}}(P_i)] \geq \Pr[R_{NN} < R_i] \geq 1 - \exp(-\alpha(d) \cdot \frac{N}{K})$, where $\alpha(d)$ is a constant dependent to d. □

As observed in Equation 22, the lower bound increases with the ratio of N to K. This shows that TSketcher is particularly suitable for our target scenario: extracting TSketches (K) from a large number of subsequences (N) from a long time series X. Furthermore, the equation also has a constant $\alpha(d)$ that penalizes high dimensionality. Thus, we operate TSketcher on the low-dimensional embedding space after the encoder in Section 3.1.2. In addition, we also prove that there exists a similar lower bound guarantee for the situation that our TSketcher is operated on the data following other distributions which can be approximated by Gaussian Mixture Models (GMMs) in Appendix A.3.

Time complexity analysis. Let N be the total number of subsequences obtained from a query time series X, and K be the number of prototypes. ① Routing: Building the index on prototypes allows each of the N subsequences [1] to be searched in $O(\log K)$, leading to a total cost of $O(N \log K)$. ② Populating: Finding the empty candidate set to populate it needs $O(K)$ time complexity. ③ Searching: For each prototype P_i, we identify its nearest subsequence s_i from the candidate set $\mathcal{T}_{P_i}$. Since $\sum_{i=1}^{K} |\mathcal{T}_{P_i}| = N$, the complexity of this step requires $O(N)$. Thus, the total time complexity of TSketcher is $O(N + K + NlogK) = O(NlogK)$.

[1] One may generate subsequences of variable lengths and extend the dist function to have a more comprehensive search of prototypes. This will increase N and hence the runtime. Importantly, to prove the concepts, our experiments show that generating N subsequences is sufficient to have a significant improvement in accuracies.

4 Experimental evaluation

We empirically evaluate TSketches for time series transformers. Section 4.1 introduces the setup and benchmarks, followed by overall results on efficiency and accuracy (Section 4.2). We then present ablation studies on StrataLoss and TSketcher (Sections 4.3–4.4), parameter sensitivity (Section 4.5), and a case study (Section 4.6) for illustrating TSketch.

4.1 Experiment Setup

Hardware and Software Environment. All experiments were executed on a server configured with a single NVIDIA V100-32G GPU and an Intel Xeon Gold 6226 CPU operating at 2.70GHz. The software stack was built upon Python 3.8, with PyTorch version 1.10.0 serving as the primary deep learning framework.

Datasets. We evaluate TSketch on the eight longest time series datasets from the UCR/UEA Time Series Archive.[2] To further investigate scalability, we also generate synthetic datasets with lengths varying systematically, ranging from 2^9 to 2^{17}. Table 3 details some dataset statistics.

Evaluation Metrics. We assess the performance of our proposed method from two perspectives: efficiency and effectiveness. For efficiency, we report the training time of each method. For effectiveness, we used time series classification, as it has comprehensive results for comparison. Our primary metric is accuracy. Following the previous works [21, 22, 50], we provide a robust evaluation by also reporting the average rank across all datasets. We also perform Friedman and Wilcoxon signed-rank tests to compare TSketch with competing baselines.

Evaluation Details. For fair comparison in training time, all baselines were run with identical hyperparameter settings in each dataset. The search of TSketcher of the training set can be considered a preprocessing step of the transformer training. Meanwhile, since its runtime is *negligible* compared to the transformer's training time, we exclude it from the training time measurements and report it separately in Section 4.3. For accuracy evaluation, we perform hyperparameter tuning by splitting the training set into 80% for training and 20% for validation, where the validation set was used to optimize transformer-specific hyperparameters. The validation set is used specifically to optimize transformer-related hyperparameters. Following prior work [6, 24], we set $Q = 4$ and $B = 2$ for iSAX, and $\alpha = 1$ for the loss function. All hyperparameter configurations are provided in the Appendix A.2.

Benchmarked Methods. To evaluate whether our TSketch can enhance the performance of existing time series transformers, we benchmark five representative models spanning the canonical transformer and several efficient variants: **Informer** [46]: Proposes ProbSparse attention to compute only the most significant query–key pairs, reducing complexity to $O(N \log N)$. **Performer** [8]: Uses random feature attention to approximate softmax attention, achieving linear complexity $O(N)$. **Fedformer** [47]: Performs attention in the frequency domain over a fixed number of Fourier components, yielding linear complexity. **DARKER** [49]: Introduces a data-driven kernel-based attention mechanism tailored for time series, with linear complexity. **Vanilla Transformer** [38]: The original architecture, widely adopted in time series applications [7, 21, 37, 44], with quadratic $O(N^2)$ complexity.

[2] www.timeseriesclassification.com

Table 3: Statistics of datasets used in our experiments.

Dataset	Train Size	Test Size	Length	Type
RightWhaleCalls	10,934	1,962	4,000	Audio
KeplerLightCurves	920	399	4,767	Sensor
FruitFlies	17,259	17,259	5,000	Audio
FaultDetectionA	10,912	2,728	5,120	Sensor
CatsDogs	138	137	14,773	Audio
BinaryHeartbeat	204	205	18,530	Audio
UrbanSound	2,713	2,712	44,100	Audio
DucksAndGeese	50	50	236,784	Audio
Synthetic	1,000	1,000	2^9–2^{17}	Generated

4.2 Overall Evaluation

We conduct a comprehensive evaluation of TSketch on its efficiency (Section 4.2.1), scalability (Section 4.2.2), and accuracy (Section 4.2.3), demonstrating its superior performance over baselines.

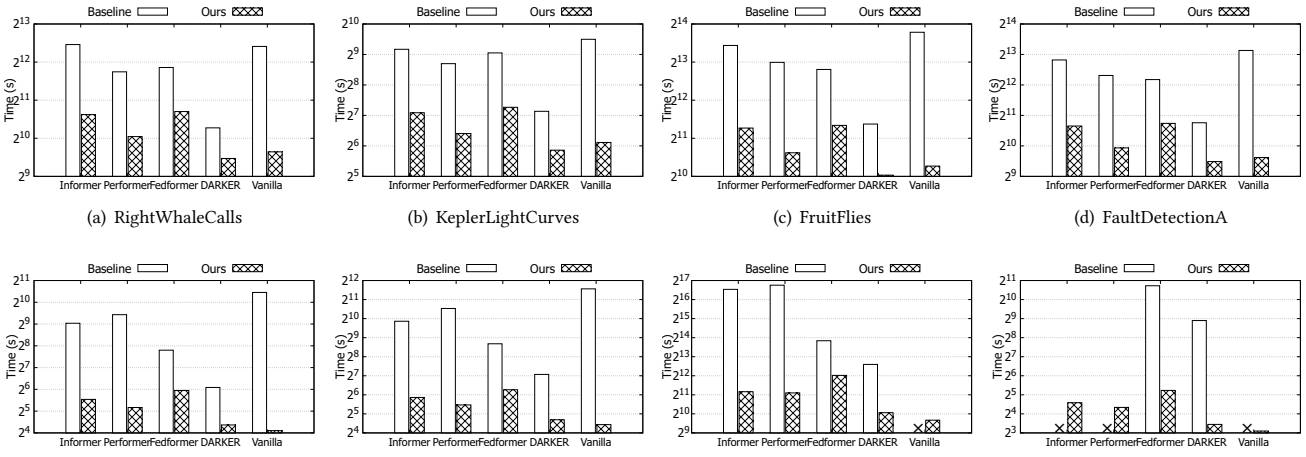
4.2.1 Efficiency evaluation We evaluate the training efficiency of TSketch by reporting the end-to-end training time across all datasets. For each backbone model, we compare two input settings: (1) using raw time series values (*Baseline*) and (2) using TSketch tokens (*Ours*). Figure 10 presents the results, ordered by ascending sequence length N.

Across all datasets, TSketch consistently reduces training time by achieving speedups of 2×–120× compared to the baselines. For example, on *DucksAndGeese*, Fedformer requires more than 2^{10} seconds to train, whereas with TSketch the training time drops below 2^5 seconds. Further, we observe that the magnitude of acceleration varies depending on the base model's complexity. For instance, TSketch accelerates the computationally intensive vanilla transformer by 7×–120×. Even for SOTA efficient models like DARKER, TSketch still provides a 2×–40× speed up. This also demonstrates TSketch's broad compatibility, as it can be effectively integrated with both traditional and highly optimized transformer models to further boost their efficiency.

The speedup becomes more pronounced as the length of the time series increases. For example, with the vanilla Transformer, the acceleration scales from 7× on RightWhaleCalls ($N = 4,767$) to 120× on BinaryHeartBeat ($N = 18,530$), underscoring our method's strength in handling long time series. TSketch not only accelerates training but also enables models to run in scenarios where baselines fail due to GPU memory overflow. Notably, Informer and vanilla Transformers encounter GPU OOM errors on *DucksAndGeese* and *UrbanSound*, but their TSketch-enhanced versions successfully complete training.

Overall, these results demonstrate that TSketch provides both efficiency and scalability: it not only reduces training time substantially but also extends the applicability of transformer-based models to previously infeasible long-sequence datasets.

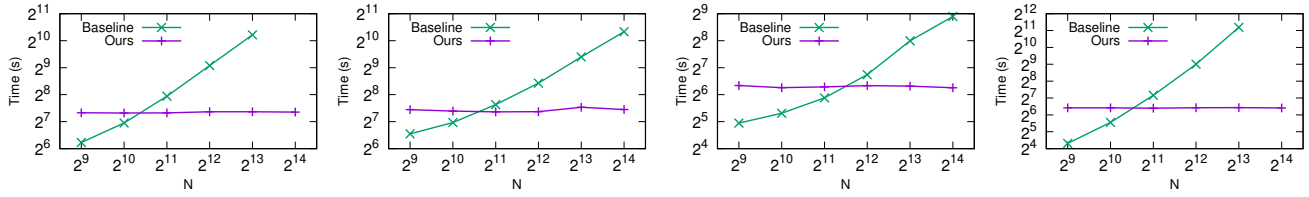
4.2.2 Scalability analysis To further investigate scalability, we evaluate the training time of using raw time series values and TSketch on synthetic datasets with input lengths N systematically varied from 2^9 to 2^{14}.



(a) RightWhaleCalls (b) KeplerLightCurves (c) FruitFlies (d) FaultDetectionA

(e) CatsDogs (f) BinaryHeartbeat (g) UrbanSound (h) DucksAndGeese

Figure 10: Training time on 8 datasets. The symbol "×" denotes a GPU out-of-memory (OOM) error, which occurred when a model could not process an excessively long input sequence.



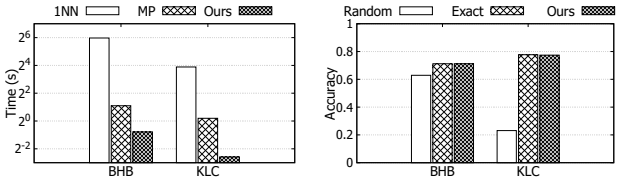
(a) Informer (b) Fedformer (c) DARKER (d) Vanilla transformer

Figure 11: Training time on varying the input length N on the synthetic dataset

From Figure 11, we can observe that the training time for the baseline models grows with the input length N, consistent with their $O(N)$ or higher computational complexity. In contrast, in our method, we compute the K TSketches for the time series as input to the transformer. Consequently, the training time becomes independent of N and remains almost constant across all datasets, demonstrating our method's ability to handle arbitrarily long time series datasets with a fixed computational budget.

4.2.3 Accuracy results Table 4 reports classification accuracy on nine UCR datasets using five transformer backbones. For each backbone, we also compare the vanilla baseline with its enhanced version using TSketch. Several consistent observations can be drawn.

First, across almost all datasets and backbones, TSketch leads to substantial improvements over the baseline. For example, on *FaultDetectionA*, the accuracy of the vanilla Transformer rises from 0.457 to 0.948, and similar gains are observed for Informer (from 0.447 to 0.931) and Fedformer (from 0.448 to 0.901). This demonstrates the effectiveness of using TSketch as opposed to values as tokens. Second, we can observe that TSketch improves the average rank of all baselines. For instance, Informer with TSketch improves its average rank from 7.188 to 2.812, and Fedformer from 8.062 to 2.938. The consistent rank gains indicate that TSketch acts as a general plug-in module, boosting diverse transformer variants. Third, TSketch often enables a backbone to achieve the best result on a dataset. For example, on *CatsDogs*, DARKER+TSketch achieves the top accuracy



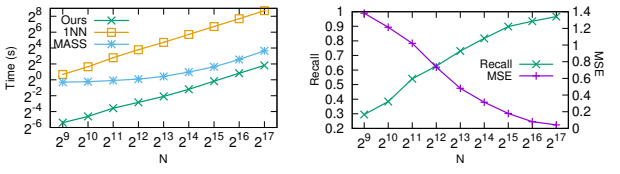
(a) Search Time (b) Accuracy

Figure 12: Performance of TSketcher in real datasets.

of 0.701, while on *KeplerLightCurves*, Informer+TSketch achieves 0.779, outperforming all other models. In addition, the statistical tests (last row) confirm that the improvements are significant, with p-values below 0.05 for all five backbones. Taken together, these results validate that TSketch consistently improves both accuracy and ranking, and can be seamlessly integrated into existing transformer frameworks for time series analysis. Finally, we find that using Vanilla transformer+TSketch has the best average rank. Given that its training time is also highly competitive, often faster than or comparable to other methods when using TSketch (as shown in Figure 10), we can observe that Vanilla Transformer+TSketch is an effective and efficient solution for long time series datasets.

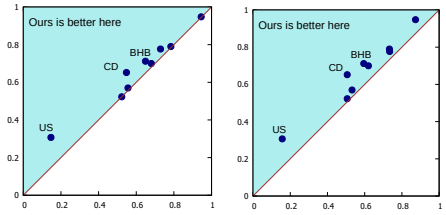
Table 4: Accuracy of our method and benchmarked methods on 8 UCR/UEA datasets. Each transformer baseline is compared with its enhanced version using TSketch (denoted as "Ours"). Best results per dataset are highlighted in bold.

Dataset	Informer		Performer		Fedformer		DARKER		Vanilla Transformer	
	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours	Baseline	Ours
RightWhaleCalls	0.503	0.507	0.501	0.504	0.506	0.508	0.501	0.508	**0.532**	0.523
KeplerLightCurves	0.391	**0.779**	0.303	0.704	0.318	0.764	0.351	0.774	0.474	0.777
FruitFlies	0.557	0.787	0.544	0.762	0.545	**0.797**	0.625	0.781	0.674	0.790
FaultDetectionA	0.447	0.931	0.424	0.889	0.448	0.901	0.446	0.906	0.457	**0.948**
CatsDogs	0.500	0.646	0.445	0.554	0.482	0.634	0.500	**0.701**	0.494	0.652
BinaryHeartbeat	0.707	0.726	0.668	0.731	0.683	**0.731**	0.717	0.727	0.697	0.712
UrbanSound	0.122	0.156	0.123	0.124	0.133	0.153	0.142	0.130	N/A	**0.307**
DucksAndGeese	N/A	0.680	N/A	0.480	0.240	0.680	0.240	0.640	N/A	**0.700**
Average Rank	7.188	2.812	9.188	5.438	8.062	2.938	7.312	3.438	6.500	2.125
p-value	—	0.008	—	0.039	—	0.008	—	0.039	—	0.016



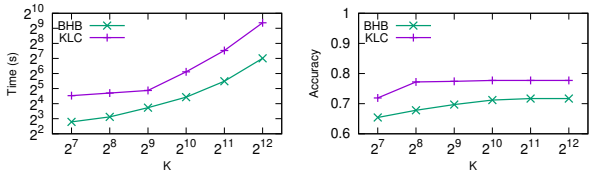
(a) Search Time (b) Recall and MSE

Figure 13: Performance of TSketcher varying input lengths on the synthetic data.



(a) With (ours) vs. without kernels (b) Ours vs. triplet loss

Figure 14: Accuracy comparison on Strataloss



(a) Training time on varying K (b) Accuracy on varying K

Figure 15: Performance on varying K

4.3 Evaluation of TSketcher

We evaluate the efficiency, accuracy, and scalability of the proposed TSketcher using two representative datasets: *BinaryHeartbeat* (BHB) and *KeplerLightCurves* (KLC).

4.3.1 Efficiency of TSketcher To measure efficiency, we record the time required to search TSketch for each prototype given an input time series. We compare against two baselines: (i) direct 1-NN search, the naive approach used in existing shape-based transformers [21, 49, 50], and (ii) MASS [45], a widely adopted method for fast subsequence matching. As shown in Figure 12(a), TSketcher achieves up to 4× speedup over MASS and over 60× compared with 1-NN.

4.3.2 Accuracy of TSketcher We compare TSketcher with two alternatives: (i) randomly selecting K subsequences from the time series as input; and (ii) taking the exact nearest neighbor of each prototype as input. Figure 12(b) shows that in both datasets, our approach matches the accuracy of exact search, and consistently outperforms random selection, demonstrating its ability to efficiently identify TSketch without compromising accuracy.

4.3.3 Scalability Analysis of TSketcher To further investigate scalability, we use synthetic datasets with input lengths N ranging from 2^9 to 2^{17}. As illustrated in Figure 13(a), the search time of TSketcher grows more slowly than 1-NN and MASS. At $N = 2^{17}$, it is more than two orders of magnitude faster than 1-NN and nearly one order faster than MASS. Figure 13(b) shows the recall and MSE of TSketcher results compared to the exact 1-NN ground truth. As N increases, the recall of TSketcher improves, approaching 1.0 (indicating an exact search) at 2^{17}. Concurrently, the MSE decreases as N increases. Both results demonstrate that TSketcher provides accurate searching of TSketch, particularly for long time series.

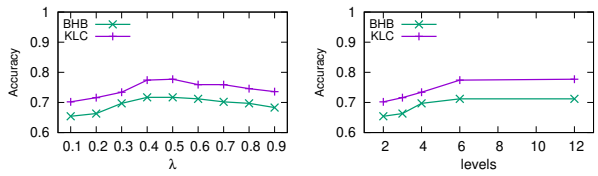
4.4 Experiment on Strataloss

We then conduct ablation studies to evaluate the effectiveness of Strataloss, with results illustrated in Figure 14. Figure 14(a) compares Strataloss with and without the convolution kernel term $\mathcal{L}_0$, showing consistent accuracy gains across datasets. This confirms that kernels help capture multi-scale variations and alleviate issues related to prototype scaling. Figure 14(b) compares Strataloss against the vanilla triplet loss, where the stratified margins lead to clear improvements by preserving relative similarity. Meanwhile, we highlight the three datasets with the most significant accuracy gains, *UrbanSound* (US), *CatsDogs* (CD), and *BinaryHeartbeat* (BHB), all of which have time series lengths exceeding 10,000. This suggests that Strataloss is particularly effective for handling long time series. Overall, both studies demonstrate that Strataloss produces a robust and discriminative embedding space.

4.5 Parameter Sensitivity

In this subsection, we study the effect of three key parameters of Tstudio on BHB and KLC.

- **Number of TSketches (K).** Figure 15 shows that larger K substantially increases training cost, while accuracy stabilizes



(a) Accuracy on varying λ (b) Accuracy on varying the number of strata

Figure 16: Parameter of StrataLoss

around $K = 1024$ for both datasets. We set $K = 1024$ by default, though other values may be tuned for specific applications.

- **Balancing weight (λ) in Equation 19.** As shown in Figure 16(a), accuracy peaks near $\lambda = 0.5$, confirming the benefit of jointly optimizing StrataLoss and reconstruction loss.
- **Number of strata (m).** Figure 16(b) indicates that fewer strata degrade accuracy, while more strata consistently improve it. This validates the importance of multi-level stratification for learning richer representations.

4.6 A Case Study on BinaryHeartbeat

To validate that our proposed TSketch effectively captures semantic information, we conduct a case study on the *BinaryHeartbeat* dataset, with heart sound recordings labeled as *normal* or *abnormal*. Each recording is a univariate time series of length 18,530. We set $K = 128$ and the sliding window sizes as $[300, 600, 900]$.

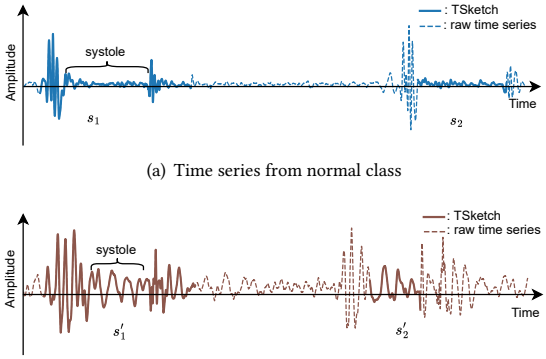
We randomly select one sample from each class. The raw time series is shown as dashed lines in Figure 17, while the solid lines represent the TSketch matched from a same prototype using TSketcher. First, for TSketch s_1 and s'_1, they nearly align with the two heart sounds in a heartbeat cycle, serving as a semantic unit used by clinicians when diagnosing cardiac conditions. Meanwhile, the TSketches of two samples show a marked divergence in *systole* (heart contraction phase): the heart sound amplitudes for the normal sample are low and stable, while those of the abnormal sample are high and volatile. These findings demonstrate that our proposed TSketch not only carries rich semantic information but also reduces redundancy by discarding repeated cycles, thus improving both the efficiency and accuracy of the transformers. In addition, for TSketches s_2 and s'_2, they carry the same semantic meaning, but are different at scales, which illustrates that the $\mathcal{L}_{sc}$ in StrataLoss successfully handles the *global scaling*.

5 Related Work

In this section, we first briefly review the existing transformers for time series, especially the efficient ones. Furthermore, we report the transformers that operate on time series subsequences, highlighting their unique representational advantages.

5.1 Efficient Transformers for Time Series

Transformer-based models have been widely used for time series, demonstrating superior performance in various tasks, including classification [21, 44, 50], imputation [16, 20, 43], and forecasting [7, 29, 46]. Despite their success, these models inherently have the quadratic complexity of the attention mechanism [34, 40].



(a) Time series from normal class

(b) Time series from abnormal class

Figure 17: TSketches of two time series from BinaryHeartbeat.

A large body of work has focused on improving the efficiency of time series transformers. Informer [46] introduces a ProbSparse attention mechanism that exploits the sparsity of attention weights, reducing the complexity of long-sequence forecasting to $O(N \log N)$. Fedformer [47] computes the attention in the frequency domain via Fourier and wavelet transforms, achieving linear complexity by operating on a fixed subset of frequency components. Despite these advances, existing methods share a fundamental limitation: their input tokens are raw values at each timestamp. As a result, the input sequence length N remains unchanged, which subsequently adversely affects the efficiency, especially for long time series.

5.2 Shape-based Transformers

Recent research explores tokenizing time series into semantic subsequences, rather than individual timestamps, to improve transformer performance [26, 27, 50]. PatchTST [29] adopts fixed-size patches as tokens, but its rigid windowing can misalign with intrinsic patterns. SVP-T [50] selects representative subsequences (shapes) near cluster centroids to serve as tokens. DARKER [49] proposes a data-driven kernel-based attention mechanism, tailoring attention to shape information in time series. Shapeformer [21] discovers shapelets offline and feeds the best-matching subsequences into the transformer. Despite these advances, the process of selecting representative subsequences is computationally costly and does not consider the global scaling.

6 Conclusion

In this paper, we propose TSketch, a new time series primitive for efficient transformer models, and Tstudio, a framework that extracts them. Tstudio first organizes subsequences via iSAX-LSH and an iSAX bucket distance with a probabilistic order-preserving (POP) guarantee, then learns representations with StrataLoss, and finally uses TSketcher to select the approximate best matching per prototype as TSketches. Our experiments on real and synthetic datasets, and five transformer backbones, replacing value tokens with TSketches significantly reduce training time for time series transformers and lead to superior or highly competitive accuracy. In the future, we plan to further improve the efficiency of time series transformers by reducing the I/O time of loading TSketches onto the GPU.

References

[1] Gérard Biau and Luc Devroye. 2015. *Lectures on the nearest neighbor method*. Vol. 246. Springer.

[2] Christopher M Bishop and Nasser M Nasrabadi. 2006. *Pattern recognition and machine learning*. Vol. 4. Springer.

[3] Paul Boniol, Mohammed Meftah, Emmanuel Remy, and Themis Palpanas. 2022. dcam: Dimension-wise class activation map for explaining multivariate data series classification. In *ACM SIGMOD*. 1175–1189.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901.

[5] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NeurIPS 2020)*. Curran Associates Inc., Red Hook, NY, USA, Article 159, 25 pages.

[6] Alessandro Camerra, Themis Palpanas, Jin Shieh, and Eamonn Keogh. 2010. isax 2.0: Indexing and mining one billion time series. In *2010 IEEE International Conference on Data Mining*. IEEE, 58–67.

[7] Yunyao Cheng, Chenjuan Guo, Bin Yang, Haomin Yu, Kai Zhao, and Christian S. Jensen. 2024. A Memory Guided Transformer for Time Series Forecasting. *Proc. VLDB Endow.* 18, 2 (2024), 239–252.

[8] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2021. Rethinking attention with performers. In *ICLR*.

[9] Jonathan H. Clark, Dan Garrette, Iulia Turc, and John Wieting. 2022. Canine: Pre-training an Efficient Tokenization-Free Encoder for Language Representation. *Transactions of the Association for Computational Linguistics* 10 (2022), 73–91.

[10] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116* (2019).

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

[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

[13] 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. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *ICLR*.

[14] Karima Echihabi. 2020. High-Dimensional Vector Similarity Search: From Time Series to Deep Network Embeddings. In *ACM SIGMOD*, David Maier, Rachel Pottinger, AnHai Doan, Wang-Chiew Tan, Abdussalam Alawini, and Hung Q. Ngo (Eds.). 2829–2832.

[15] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. *Advances in neural information processing systems* 32 (2019).

[16] Ju-Sheng Hong, Junwen Yao, Jonas Mueller, and Jane-Ling Wang. 2024. SAND: Smooth imputation of sparse and noisy functional data with Transformer networks. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

[17] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data mining and knowledge discovery* 33, 4 (2019), 917–963.

[18] 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. 2020. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (2020), 1936–1962.

[19] Duc Kieu, Tung Kieu, Peng Han, Bin Yang, Christian S Jensen, and Bac Le. 2024. TEAM: Topological Evolution-aware Framework for Traffic Forecasting–Extended Version. *Proc. VLDB Endow.* 18, 2 (2024), 265–278.

[20] Zhichen Lai, Dalin Zhang, Huan Li, Dongxiang Zhang, Hua Lu, and Christian S. Jensen. 2024. ReCTSi: Resource-efficient Correlated Time Series Imputation via Decoupled Pattern Learning and Completeness-aware Attentions. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (Barcelona, Spain) *(KDD '24)*. Association for Computing Machinery, New York, NY, USA, 1474–1483. https://doi.org/10.1145/3637528.3671816

[21] Xuan-May Le, Ling Luo, Uwe Aickelin, and Minh-Tuan Tran. 2024. Shapeformer: Shapelet transformer for multivariate time series classification. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 1484–1494.

[22] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace LH Wong. 2021. Shapenet: A shapelet-neural network approach for multivariate time series classification. In *AAAI*. 8375–8383.

[23] Guozhong Li, Byron Choi, Jianliang Xu, Sourav S Bhowmick, Kwok-Pan Chun, and Grace Lai-Hung Wong. 2020. Efficient shapelet discovery for time series classification. *IEEE transactions on knowledge and data engineering* 34, 3 (2020), 1149–1163.

[24] Guozhong Li, Byron Choi, Rundong Zuo, Sourav S Bhowmick, and Jianliang Xu. 2025. leSAX Index: A Learned SAX Representation Index for Time Series Similarity Search. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*. IEEE Computer Society, 1995–2008.

[25] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15 (2007), 107–144.

[26] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. 2024. iTransformer: Inverted Transformers Are Effective for Time Series Forecasting. In *The Twelfth International Conference on Learning Representations*. https://openreview.net/forum?id=JePfAI8fah

[27] Yong Liu, Guo Qin, Xiangdong Huang, Jianmin Wang, and Mingsheng Long. 2025. Timer-XL: Long-Context Transformers for Unified Time Series Forecasting. In *The Thirteenth International Conference on Learning Representations*. https://openreview.net/forum?id=KMCJXjlDDr

[28] Navid Mohammadi Foumani, Lynn Miller, Chang Wei Tan, Geoffrey I Webb, Germain Forestier, and Mahsa Salehi. 2024. Deep learning for time series classification and extrinsic regression: A current survey. *Comput. Surveys* 56, 9 (2024), 1–45.

[29] Yuqi Nie, Nam H. Nguyen, Phanwadee Sinthong, and Jayant Kalagnanam. 2023. A Time Series is Worth 64 Words: Long-term Forecasting with Transformers. In *ICLR*.

[30] Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. 2021. Random Feature Attention. In *ICLR*.

[31] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* 21, 140 (2020), 1–67.

[32] Jin Shieh and Eamonn Keogh. 2008. iSAX: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. 623–631.

[33] Yuxin Tang, Feng Zhang, Jiawei Guan, Yuan Tian, Xiangdong Huang, Chen Wang, Jianmin Wang, and Xiaoyong Du. 2025. Improving Time Series Data Compression in Apache IoTDB. *Proceedings of the VLDB Endowment* 18, 10 (2025), 3406–3420.

[34] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *Comput. Surveys* 55, 6 (2022), 1–28.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971* (2023).

[36] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aur'elien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *ArXiv* abs/2302.13971 (2023). https://api.semanticscholar.org/CorpusID:257219404

[37] Shreshth Tuli, Giuliano Casale, and Nicholas R. Jennings. 2022. TranAD: Deep Transformer Networks for Anomaly Detection in Multivariate Time Series Data. *PVLDB* (2022), 1201–1214.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

[39] Qitong Wang and Themis Palpanas. 2021. Deep Learning Embeddings for Data Series Similarity Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 1708–1716.

[40] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in time series: A survey. *IJCAI* (2023).

[41] Haoran Xiong, Hang Zhang, Zeyu Wang, Zhenying He, Peng Wang, and X. Sean Wang. 2024. CIVET: Exploring Compact Index for Variable-Length Subsequence Matching on Time Series. *Proc. VLDB Endow.* 17, 9 (May 2024), 2123–2135.

[42] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval. In *International Conference on Learning Representations*.

[43] Jingwen Xu, Fei Lyu, and Pong C. Yuen. 2023. Density-Aware Temporal Attentive Step-wise Diffusion Model For Medical Time Series Imputation. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management, CIKM 2023, Birmingham, United Kingdom, October 21-25, 2023*, Ingo Frommholz, Frank Hopfgartner, Mark Lee, Michael Oakes, Mounia Lalmas, Min Zhang, and Rodrygo L. T. Santos (Eds.). ACM, 2836–2845.

[44] George Zerveas, Srideepika Jayaraman, Dhaval Patel, Anuradha Bhamidipaty, and Carsten Eickhoff. 2021. A Transformer-Based Framework for Multivariate Time Series Representation Learning. In *ACM SIGKDD*. 2114–2124.

[45] Sheng Zhong and Abdullah Mueen. 2024. MASS: distance profile of a query over a time series. *Data Min. Knowl. Discov.* 38, 3 (Feb. 2024), 1466–1492.

[46] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*. 11106–11115.

[47] Tian Zhou, Ziqing Ma, Qingsong Wen, Xue Wang, Liang Sun, and Rong Jin. 2022. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In *International conference on machine learning*. PMLR, 27268–27286.

[48] Zhihao Zhuang, Yingying Zhang, Kai Zhao, Chenjuan Guo, Bin Yang, Qingsong Wen, and Lunting Fan. 2024. Noise matters: Cross contrastive learning for flink anomaly detection. *Proceedings of the VLDB Endowment* 18, 4 (2024), 1159–1168.

[49] Rundong Zuo, Guozhong Li, Rui Cao, Byron Choi, Jianliang Xu, and Sourav S Bhowmick. 2024. DARKER: Efficient Transformer with Data-driven Attention Mechanism for Time Series. *Proceedings of the VLDB* 17, 11 (2024), 3329–3242.

[50] Rundong Zuo, Guozhong Li, Byron Choi, Sourav S Bhowmick, Daphne Ngai-Yin Mah, and Grace Lai-Hung Wong. 2023. SVP-T: A Shape-Level Variable-Position Transformer for Multivariate Time Series Classification. In *AAAI*. 11497–11505.

A Appendix

A.1 Proof of Theorem: Probabilistic Order-Preserving Property of iSAX Bucket Distance

Definition 2. ***(iSAX distance (MinDist) [25])*** *Given two time series subsequences $T = (t_1, \ldots, t_n), U = (u_1, \ldots, u_n)$, and their corresponding iSAX representations $b_T = (b_T^1, \ldots, b_T^w), b_U = (b_U^1, \ldots, b_U^w)$, where w is the number of PAA segments. Assume each pair b_T^k and b_U^k maps to value intervals $[x_t, y_t]$ and $[x_u, y_u]$, respectively. The distance between the k-th segments is defined as:*

$$dist_i(b_T^k, b_U^k) = \begin{cases} 0, & \text{if } b_T^k = b_U^k \\ \min(|x_t - y_u|, |x_u - y_t|), & \text{else.} \end{cases} \tag{30}$$

Then the distance between two iSAX representations is computed as:

$$MinDist(b_T, b_U) \equiv \sqrt{\frac{n}{w}} \sqrt{\sum_{i=1}^{w} \left(dist(b_T^k, b_U^k)\right)^2} \tag{31}$$

Theorem A.1 (**Probabilistic Order-Preserving Property of iSAX Bucket Distance**). *Let w be a positive integer, and let $T = (t_1, \ldots, t_n)$ and $U = (u_1, \ldots, u_n)$ be two time series subsequences such that $t_1, \ldots, t_n, u_1, \ldots, u_n \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1)$.*

Assume their iSAX Bucket Distance is $iBDIST(b_T, b_U) = d_{iB}$. Then, for any $\epsilon, \delta > 0$, the following inequality holds:

$$\begin{aligned} &\Pr[\|T - U\|_2^2 \le n\delta^2 + d_{iB}\frac{n}{w}(d_z^2 + 2\delta d_z) + 2(n - w) + \epsilon] \\ &\ge (2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w - d_{iB}} \\ &\cdot \frac{\epsilon^2}{8n(1 - \frac{w}{n})^2 + \epsilon^2} \end{aligned} \tag{32}$$

where $d_z = \Phi^{-1}(1 - \frac{1}{2^B}) - \Phi^{-1}(1 - \frac{2}{2^B})$ is a constant, $\Phi(\cdot)$ is CDF of the standard Gaussian distribution and B is the number of iSAX bits per segment.

Proof.

$$\begin{aligned} \|T - U\|_2^2 &= \sum_{i=1}^{w}\sum_{j=1}^{\frac{n}{w}}(t_{ij} - u_{ij})^2 \\ &= \sum_{i=1}^{w}\{\sum_{j=1}^{\frac{n}{w}}[(t_{ij} - \bar{t_i}) - (u_{ij} - \bar{u_i}) + (\bar{t_i} - \bar{u_i})]^2\} \\ &= \frac{n}{w}\sum_{i=1}^{w}(\bar{t_i} - \bar{u_i})^2 + \sum_{i=1}^{w}\sum_{j=1}^{\frac{n}{w}}[(t_{ij} - \bar{t_i}) - (u_{ij} - \bar{u_i})]^2 \end{aligned} \tag{33}$$

Let $A = \frac{n}{w}\sum_{i=1}^{w}(\bar{t_i} - \bar{u_i})^2$ and $G = \sum_{i=1}^{w}\sum_{j=1}^{\frac{n}{w}}[(t_{ij} - \bar{t_i}) - (u_{ij} - \bar{u_i})]^2$. Assume $iBDIST(b_T, b_U) = d_{iB}$ and it indicates that at most d_{iB} segments in the iSAX representations of T and U are different. Under the circumstance, we first calculate $|\bar{t_i} - \bar{u_i}|$ where t_i and u_i lie in adjacent interval in each segment. Let $\{z_i\}_{i=1}^{2^B-1}$ be the quantile points in each segment and assume $|\bar{t_i} - \bar{u_i}| \le |z_{2^B-1} - z_{2^B-2}| + \delta$ where $\delta > 0$. According to the property of iSAX, $d_z = |z_{2^B-1} - z_{2^B-2}| = \Phi^{-1}(1 - \frac{1}{2^B}) - \Phi^{-1}(1 - \frac{2}{2^B})$ where $\Phi(\cdot)$ is the cumulative distribution function of the Gaussian distribution. Additionally, since $\bar{t_i} - \bar{u_i} \sim \mathcal{N}(0, \frac{2w}{n})$, $\forall \delta > 0$, we have the inequality below:

$$\Pr[|\bar{t_i} - \bar{u_i}| \le d_z + \delta] = 2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1 \tag{34}$$

Therefore, we can calculate A:

$$\begin{aligned} A &= \frac{n}{w}\sum_{i=1}^{w}(\bar{t_i} - \bar{u_i})^2 \\ &\le D_A(d_{iB}, \delta) = \frac{n}{w}(d_{iB} \cdot (d_z + \delta)^2 + (w - d_{iB}) \cdot \max(\delta, d_z)^2) \\ &= n \cdot \max(\delta, d_z)^2 + d_{iB} \cdot \frac{n}{w}(\min(\delta, d_z)^2 + 2\delta d_z) \end{aligned} \tag{35}$$

There exists:

$$\begin{aligned} &\Pr[A \le n \cdot \max(\delta, d_z)^2 + d_{iB} \cdot \frac{n}{w}(\min(\delta, d_z)^2 + 2\delta d_z)] \\ &\ge (2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w - d_{iB}} \end{aligned} \tag{36}$$

Then we analyze the distribution of G. Since $t_{ij} \sim \mathcal{N}(0, 1)$, we obtain $\mathbb{E}[t_{ij} - \bar{t_i}] = 0$ and $\mathrm{Var}(t_{ij} - \bar{t_i})$ as follows.

$$\begin{aligned} \mathrm{Var}(t_{ij} - \bar{t_i}) &= \mathrm{Var}(t_{ij}) + \mathrm{Var}(\bar{t_i}) - 2\mathrm{Cov}(t_{ij}, \bar{t_i}) \\ &= 1 + \mathrm{Var}(\frac{w}{n}\sum_{k=1}^{\frac{n}{w}} t_{ik}) - 2\mathrm{Cov}(t_{ij}, \frac{w}{n}\sum_{k=1}^{\frac{n}{w}} t_{ik}) \\ &= 1 + (\frac{w}{n})^2\sum_{k=1}^{\frac{n}{w}}\mathrm{Var}(t_{ik}) - 2\frac{w}{n}\sum_{k=1}^{\frac{n}{w}}\mathrm{Cov}(t_{ij}, t_{ik}) \\ &= 1 + (\frac{w}{n})^2 \cdot \frac{n}{w} - 2\frac{w}{n} = 1 - \frac{w}{n} \end{aligned} \tag{37}$$

Thus it yields $(t_{ij} - \bar{t_i}), (u_{ij} - \bar{u_i}) \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1 - \frac{w}{n})$. Let $z_{ij} = (t_{ij} - \bar{t_i}) - (u_{ij} - \bar{u_i})$, it holds $z_{ij} \sim \mathcal{N}(0, 2(1 - \frac{w}{n}))$. Then $\sum_{j=1}^{\frac{n}{w}} \frac{z_{ij}^2}{2(1-\frac{w}{n})} \sim \chi^2(\frac{n}{w})$ and $\frac{B}{2(1-\frac{w}{n})} = \sum_{i=1}^{w}\sum_{j=1}^{\frac{n}{w}} \frac{z_{ij}^2}{2(1-\frac{w}{n})} \sim \chi^2(n)$. We obtain $\mathbb{E}[G] = 2(1 - \frac{w}{n}) \cdot n = 2(n - w)$ and $\mathrm{Var}(G) = 8n(1 - \frac{w}{n})^2$. According to the one-sided Chebyshev inequality, for $\forall \epsilon > 0$, it holds

$$\Pr[G - 2(n - w) \ge \epsilon] \le \frac{8n(1 - \frac{w}{n})^2}{8n(1 - \frac{w}{n})^2 + \epsilon^2}. \tag{38}$$

Conversely, we have

$$\Pr[G \le 2(n - w) + \epsilon = D_B(\epsilon)] \ge \frac{\epsilon^2}{8n(1 - \frac{w}{n})^2 + \epsilon^2} \tag{39}$$

After combining Equation 36 and 39, it yields

$$\begin{aligned} &\Pr[\|T - U\|_2^2 = A + G \le D_A(d_{iB}, \delta) + D_G(\epsilon)] \\ &\ge (2\Phi(\sqrt{\frac{n}{2w}}(d_z + \delta)) - 1)^{d_{iB}} \cdot (2\Phi(\sqrt{\frac{n}{2w}}\max(\delta, d_z) - 1))^{w - d_{iB}} \\ &\cdot \frac{\epsilon^2}{8n(1 - \frac{w}{n})^2 + \epsilon^2} \end{aligned} \tag{40}$$

□

A.2 Hyperparameter for training

Table 5 summarizes the training hyperparameters used across all datasets. To accommodate datasets of varying characteristics and sizes, we employed different batch sizes (ranging from 4 to 64) and training epochs (from 50 to 200). Larger datasets were trained

Table 5: Hyperparameters

Dataset name	Batch size	Epochs
RightWhaleCalls	64	100
KeplerLightCurves	32	200
FruitFlies	16	100
FaultDetectionA	4	100
CatsDogs	4	100
BinaryHeartbeat	4	200
UrbanSound	64	50
DucksAndGeese	32	100

with bigger batch sizes for computational efficiency, while smaller datasets utilized smaller batches with more epochs to ensure adequate learning. This tailored approach ensures stable convergence and optimal performance for each dataset.

A.3 Extra Explanation of Theorem: Lower bound of Algorithm 3

Theorem A.2. (Lower bound of Algorithm 3) *Given time series prototypes* $\mathcal{P} = \{P_1, \ldots, P_K\}$ *and time series subsequences* $\mathcal{T} = \{T_1, \ldots, T_N\}$ *where* $P_1, \ldots, P_K, T_1, \ldots, T_N \overset{\text{iid}}{\sim} \mathcal{N}(0, I_d)$ *where* I_d *denotes the* $d \times d$ *identity matrix, TSketch* s_i *returned by Algorithm 3 has the following property:*

$$\Pr\left[s_i = \mathrm{NN}_{\mathcal{T}}(P_i)\right] \;\geq\; 1 - \exp\left(-\alpha(d) \cdot \frac{N}{K}\right), \tag{41}$$

where $\mathrm{NN}_A(x)$ *denotes the nearest neighbor search (NN) of* x *in set* A *and* $\alpha(d)$ *is a positive constant dependent only on the dimension* d.

Figure 18 illustrates this phenomenon by the relationships between some time series subsequences and two prototypes. Here, $V(P_i)$ denotes the Voronoi region of prototype P_i, as defined in the proof of Theorem 3.2. When the Voronoi regions of two prototypes overlap (the white area in the figure), a query subsequence located in this intersection may be assigned to its second-nearest prototype rather than the true nearest one.

Our theorem is proved based on the condition that the prototypes and time series subsequences are independent and identically Gaussian distributed. By using a sufficient number of Gaussians and by adjusting their means and covariances as well as the coefficients in the linear combination, almost any continuous density can be approximated to arbitrary accuracy [2]. Accordingly, our probability in Equation 41 and the proof can still be utilized to obtain lower bounds under the circumstance of Gaussian mixture models. Thus, the probability proves the feasibility of our approximate nearest neighbor search method since the dataset may not conform to the specific distribution in practice.

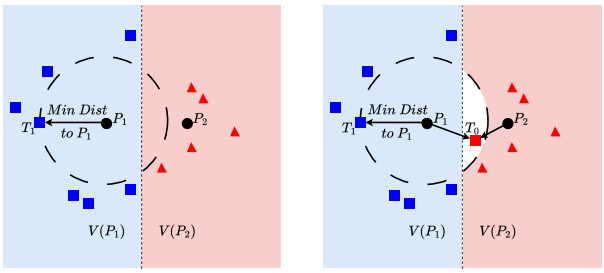
Theorem A.3. (Lower bound of Algorithm 3 of Distribution under GMM) *Assume two Gaussian mixture models* GMM_P *and* GMM_T *with their corresponding probability functions*

$$p_P(x) = \sum_{k=1}^{K_P} \pi_{P_k} \mathcal{N}(x; \boldsymbol{\mu}_{P_k}, \Sigma_{P_k}) \tag{42}$$

and

$$p_T(x) = \sum_{l=1}^{K_T} \pi_{T_l} \mathcal{N}(x; \boldsymbol{\mu}_{T_l}, \Sigma_{T_l}) \tag{43}$$

where π_{P_k}, π_{T_l} *are mixing coefficients and* $\sum_{k=1}^{K_P} \pi_{P_k} = \sum_{l=1}^{K_T} \pi_{T_l} = 1$.



(a) Two Voronoi regions (b) Two Voronoi regions with special area

Figure 18: Illustration of two example distributions on a two-dimensional surface. If one subsequence lies in the white area, our method will produce the wrong NN result.

Given time series prototypes $\mathcal{P} = \{P_1, \ldots, P_K\}$ *and time series subsequences* $\mathcal{T} = \{T_1, \ldots, T_N\}$ *where* $P_1, \ldots, P_K \overset{\text{iid}}{\sim} GMM_P$ *and* $T_1, \ldots, T_N \overset{\text{iid}}{\sim} GMM_T$. *We denote the nearest neighbor search of* x *on set* A *as* $\mathrm{NN}_A(x)$. *It holds,*

$$\Pr[s_i = \mathrm{NN}_{\mathcal{T}}(P_i)] \geq 1 - \sum_{k=1}^{K_P} \pi_{P_k} \cdot \exp\left(-\beta(d,k) \cdot \frac{\pi_{T_k}}{\pi_{P_k}} \cdot \frac{N}{K}\right) \tag{44}$$

where sketch s_i *is obtained in Algorithm 3,* β *is a constant,* N *and* K *denote the size of* $\mathcal{T}$ *and* $\mathcal{P}$.

Proof. Similar to the proof of Theorem 3.2, we still explore the probability below first:

$$\Pr[s_i = \mathrm{NN}_{\mathcal{T}}(P_i)] = \mathbb{E}_{\mathcal{T},\,\mathcal{P}}\left[\tfrac{1}{K}\textstyle\sum_{i=1}^{K} \mathbf{1}\left(\mathrm{NN}_{\mathcal{P}}\left(\mathrm{NN}_{\mathcal{T}}(P_i)\right) = P_i\right)\right] \tag{45}$$

where $\mathbf{1}$ denote the indicator function. Besides, we also have the Voronoi region of P_i denoted as $V(P_i) = \{x \in \mathbb{R}^d | \|x - P_i\| \leq \|x - P_j\|, \forall j \neq i\}$. For each P_i, we explore its nearest neighbor $T^* = \mathrm{NN}_{\mathcal{T}}(P_i)$. If $T^* \in V(P_i)$, we have $p_i = \mathrm{NN}_{\mathcal{T}}(P_i)$. Thus the probability holds,

$$\begin{aligned}
&\Pr[s_i = \mathrm{NN}_{\mathcal{T}}(P_i)] = \mathbb{E}_{\mathcal{T},\,\mathcal{P}}\left[\frac{1}{K}\sum_{i=1}^{K} \mathbf{1}\left(T^* \in V(P_i)\right)\right] \\
&= \sum_{k=1}^{K_P}\sum_{l=1}^{K_T} \Pr[P_i \in \mathcal{E}_k, T \in \mathcal{F}_l] \cdot \Pr[T^* \in V(P_i) | P_i \in \mathcal{E}_k, T \in \mathcal{F}_l] \\
&= \sum_{k=1}^{K_P}\sum_{l=1}^{K_T} \pi_{P_k}\pi_{T_l} \Pr[T^* \in V(P_i) | P_i \in \mathcal{E}_k, T \in \mathcal{F}_l]
\end{aligned} \tag{46}$$

where $\mathcal{E}_k$ denotes P comes from the k-th Gaussian component and $\mathcal{F}_l$ denotes T comes from the ℓ-th Gaussian component. If $k = \ell$, the problem degrades into the probability under a single Gaussian distribution, similar to Theorem A.2. Thus, we obtain:

$$P_{intra}^{(k)} \geq 1 - \exp\left(-\beta(d,k) \cdot \frac{N_k}{K_k}\right) \tag{47}$$

where $\beta(d,k)$ is a constant relying on dimension d and component k. N_k, K_k are the sample sizes of P, T lying into the same component, respectively. However, if $k \neq \ell$, we have:

$$P_{\text{cross}}^{(k,\ell)} = \Pr[\|P - T\| < \min_j \|P - T_j\| | P \in \mathcal{E}_k, T \in \mathcal{F}_\ell] \tag{48}$$

Then for $P \sim \mathcal{N}(\mu_{P_k}, \Sigma_{P_k}), T \sim \mathcal{N}(\mu_{T_\ell}, \Sigma_{T_\ell})$, we have $P - T \sim \mathcal{N}(\mu_{P_k} - \mu_{T_\ell}, \Sigma_{P_k} + \Sigma_{T_\ell})$. We can consider a simple case of Equation 48 where different components of GMM_P and GMM_T have

spherical symmetric covariance for analysis. In such a case, the distribution of $\|P - T\|$ in Equation 48 can be degraded into a non-central χ^2 distribution. For the sake of convenience, we present some key steps of deriving $P_{\text{cross}}^{(k,\ell)}$. We can obtain the distribution of $\|P - T\|^2$ as follows:

$$\|z\|^2 \sim (\sigma_{P_k}^2 + \sigma_{T_\ell}^2) \cdot \chi_d^2(\lambda),\ where\ \lambda = \frac{\|\mu_{P_k} - \mu_{T_\ell}\|^2}{\sigma_{P_k}^2 + \sigma_{T_\ell}^2}. \tag{49}$$

Thus, we obtain

$$\Pr[\|P - T\| \le r | P \in \mathcal{E}_k, T \in \mathcal{F}_\ell] = F_{\chi_d^2}(\frac{r^2}{\sigma_{P_k}^2 + \sigma_{T_\ell}^2}) \tag{50}$$

The function in Equation 50 is a monotonically decreasing function with respect to r^2. In other words, a larger $\mu_{P_k} - \mu_{T_\ell}$ can lead to decreasing of $P_{\text{cross}}^{(k,\ell)}$, indicating that $P_{\text{cross}}^{(k,\ell)}$ can approximate to zero with obvious separated Gaussian components of GMM_P and GMM_T when $k \neq \ell$. Therefore, we have

$$\begin{aligned}\Pr[s_i = \text{NN}_{\mathcal{T}}(P_i)] &= \sum_{k=1}^{K_P} \pi_{P_k} \pi_{T_k} P_{intra}^{(k)} + \sum_{k \neq \ell} \pi_{P_k} \pi_{T_\ell} P_{\text{cross}}^{(k,\ell)} \\ &\xrightarrow{\lambda \to \infty} \sum_{k=1}^{K_P} \pi_{P_k} \pi_{T_k} P_{intra}^{(k)},\end{aligned} \tag{51}$$

Finally, under the circumstances, we achieve

$$\Pr[s_i = \text{NN}_{\mathcal{T}}(P_i)] \ge 1 - \sum_{k=1}^{K_P} \pi_{P_k} \cdot \exp\left(-\beta(d,k) \cdot \frac{\pi_{T_k}}{\pi_{P_k}} \cdot \frac{N}{K}\right) \tag{52}$$

□