DARKER: **Efficient Transformer with Data-driven Attention Mechanism for Time Series**

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

Guozhong Li
King Abdullah University of Science and Technology
guozhong.li@kaust.edu.sa

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

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-based models have facilitated numerous applications with superior performance. A key challenge in transformers is the quadratic dependency of its training time complexity on the length of the input sequence. A recent popular solution is using random feature attention (RFA) to approximate the costly vanilla attention mechanism. However, RFA relies on only a single, fixed projection for approximation, which does not capture the input distribution and can lead to low efficiency and accuracy, especially on time series data. In this paper, we propose DARKER, an efficient transformer with a novel **DA**ta-d**R**iven **KER**nel-based attention mechanism. To precisely present the technical details, this paper discusses them with a fundamental time series task, namely, time series classification (TSC). First, the main novelty of DARKER lies in approximating the softmax kernel by learning multiple machine learning models with trainable weights as multiple projections offline, moving beyond the limitation of a fixed projection. Second, we propose a projection index (called PINDEX) to efficiently search the most suitable projection for the input for training transformer. As a result, the overall time complexity of DARKER is linear with the input length. Third, we propose an indexing technique for efficiently computing the inputs required for transformer training. Finally, we evaluate our method on 14 real-world and 2 synthetic time series datasets. The experiments show that DARKER is 3×-4× faster than vanilla transformer and 1.5×-3× faster than other SOTAs for long sequences. In addition, the accuracy of DARKER is comparable to or higher than that of all compared transformers.

PVLDB Reference Format:
Rundong Zuo, Guozhong Li, Rui Cao, Byron Choi, Jianliang Xu, and Sourav S Bhowmick. DARKER: Efficient Transformer with Data-driven Attention Mechanism for Time Series. PVLDB, 14(1): XXX-XXX, 2020.
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/darker.

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

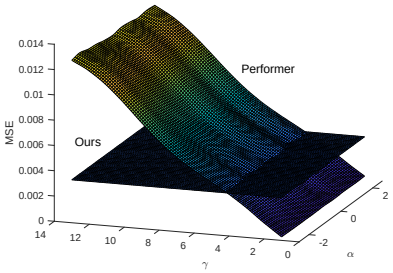


Figure 1: The approximation error of softmax in Performer [5] and our method. Two matrices, Q and K, are initialized consisting of N pairs of query and key vectors. Then, the key vector is scaled with a scalar γ and rotated with an angle α.

1 INTRODUCTION

Time series data, widely generated in various domains such as smart cities, economics, human activity recognition [18, 33], has received lots of research attention for numerous fundamental tasks, including classification [4, 8, 58], clustering [3, 34, 44], forecasting [40, 53, 57], abnormal detection [42, 50], and similarity search [11]. Recently, transformer-based methods exhibit impressive performance on time series tasks, *e.g.*, classification [58, 63] and forecasting [29, 61]. However, previous transformer-based methods [51, 61] suffer from the well-known inefficient time complexity $O(N^2)$ of attention mechanism, where N is the length of the input sequence. Many researchers have remarked on the efficiency issue of transformers, and some representative ones are quoted as follows.

- "*... vanilla Transformer has a time and memory complexity of $O(N^2)$, which becomes the computational bottleneck when dealing with long sequences.*" [51]
- "*... due to its L-quadratic computation and memory consumption on L-length inputs/outputs.*" [61]

Extensive methods have been proposed to address the efficiency issue of the transformer on image and language tasks [26, 43]. However, most of them require *priors*, such as the sparsity [61], low rank [49] of attention weight matrices, or identical input query and key matrices [17]. These prerequisites limit the applicability of relevant methods. To avoid the *priors*, nowadays, *Random Feature Attention* (RFA) methods (a.k.a *kernel-based methods*) have been proposed and become one of the well-received methods for improving the efficiency [5, 35, 37, 60]. RFA considers the $\exp(\cdot)$ function in the

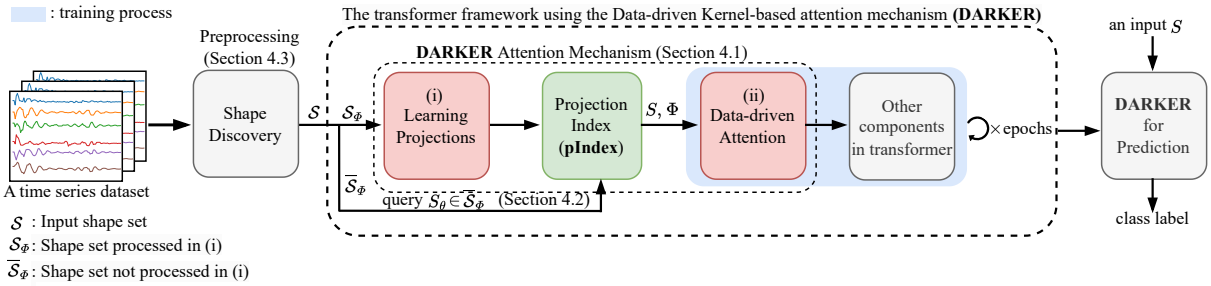


Figure 2: The overview of DARKER

attention mechanism as a kernel function and then *approximate it by a fixed projection ϕ with random features.* Although RFA methods achieve linear space and time complexity without *priors*, they face several challenges when being applied to time series analyses. For precise discussions, this paper focuses on time series classification (TSC), one of the fundamental analyses.

Challenges. A comprehensive theoretical analysis of three challenges of RFA on time series shall be detailed in Section 3. Here, we briefly highlight the challenges when applying RFA to time series. ① RFA methods require numerous computations of random features for all time series in each training epoch, which is inefficient when training a transformer model. ② Computing $\exp(\frac{\|x\|^2}{2})$ for time series in RFA methods may lead to *overflow*[1], where x is an input vector. Attributed to the high dimensionality and variance inherently in time series data, the L2-norm of x can be large, and hence, the exponential term could become extremely large. This phenomenon has been observed in various datasets, *e.g.*, BasicMotions, Cricket, and Phoneme in Section 5.3. ③ RFA methods, while claiming to be an unbiased estimator of $\exp(\cdot)$, are *not an unbiased estimator of the entire softmax in the attention mechanism* [60]. This results in a high mean square error (MSE) when approximating attention weights, which may be particularly problematic in most cases of time series data with significant variances. As illustrated in Figure 1, the MSE in Performer [5] (a representative RFA method) increases substantially as γ increases.

Our solution. In this paper, we propose an efficient transformer, DARKER (overview shown in Figure 2), whose novelty mainly lies in a DAta-dRiven KERnel-based attention mechanism. DARKER learns and indexes *multiple projections*, rather than a single fixed projection used in previous kernel-based methods.

First, in DARKER, we propose to utilize machine learning models trained from time series instances, as data-driven projections for approximating softmax, instead of directly using $\exp(\cdot)$ as kernel function and repeatedly applying a fixed projection to approximate softmax for all time series instances. Specifically, DARKER attention mechanism consists of two stages: (i) learning projections and (ii) data-driven attention. (i) In the stage of learning projections, since it is difficult to train a single machine learning model to approximate softmax for entire time series data, we separately train an additional model for each input S, where $S \in S_\Phi$. These models share a uniform structure, yet each is trained on an individual time series instance (shown in Figure 3). The model with its learned weights is regarded as the projection ϕ. As such, each time series instance is paired with its specific projection and stored in a hashmap I. (ii) In the data-driven attention stage for training a transformer model, we retrieve the projection for each time instance from I. Then, the projection is utilized to approximate and replace softmax for S. Therefore, DARKER reduces the original time complexity to $O(N)$. We remark that since the projections are learned from the input data, DARKER has a lower approximating error in handling most real-time series data than one fixed data-agnostic projection.

Second, when training a transformer, it is natural to encounter an input S_θ whose projection is not learned in the first stage ($S_\theta \in \overline{\mathcal{S}}_\Phi$). Instead of inefficiently learning S_θ's projection in the second stage, we further propose a projection index (called PINDEX) to efficiently search its projection based on existing I. Third, discriminative time series subsequences, or simply *shapes*, are crucial for training models of specific tasks such as TSC, but their discovery is computationally costly (*e.g.*, [13, 21]). We conduct a preprocessing to discover shapes S as the input of DARKER and propose a cluster-to-distance index, called I_{c2d}, to optimize its efficiency.

We conduct extensive experiments on both 14 real-world and 2 large synthetic time series datasets. The result shows that DARKER is about 3×-4× faster than the vanilla transformer and 1.5×-3× faster than the other compared methods, including RFA methods and SOTA efficient transformer for time series. Our contributions are summarized as follows:

- We propose DARKER, an efficient transformer with a data-driven kernel-based attention mechanism, which learns multiple projections for time series instead of one fixed projection. Thus, DARKER reduces time complexity from quadratic to linear.
- In DARKER, we propose an index named PINDEX for efficiently retrieving the learned projection for time series.
- We optimize shape discovery through building an I_{c2d} to improve efficiency of transformer training.
- Extensive experiments on both 14 real and 2 synthetic time series datasets show the efficiency and effectiveness of DARKER, especially for long input sequences.

Organization. The rest of this paper is organized as follows. We introduce the preliminaries in Section 2. The analysis of the random feature attention is presented in Section 3. The details of our DARKER are given in Section 4. Section 5 reports the experimental results. Section 6 reviews the related works. Section 7 concludes the paper and presents our future work.

[1]Python reports *overflow* when a value is larger than the upper bound 2^{1024}.

2 BACKGROUND

This section previews the terminologies used in the paper. Table 1 summarizes some frequently used notations and their meanings.

Time series instance. A time series instance (or simply *time series*) is denoted as $X = \{x_1, x_2, \cdots, x_t\}$, which records the events of t timestamps. For a univariate time series (UTS), x represents a single value, whereas in a multivariate time series (MTS), x is a vector of v variables. A time series dataset $\mathcal{D}$ consists of $\mathbb{N}$ time series.

Time series subsequence. A time series subsequence T is denoted as $\{x_p, x_{p+1}, \ldots, x_q\}$, where $1 \leq p \leq q \leq t$. Time series subsequences can be generated by applying a sliding window technique to a time series X.

Time series classification (TSC). Given a time series dataset $\mathcal{D}$ consists of $\mathbb{N}$ time series instances. For each time series X_i, $i \in [1, \mathbb{N}]$ has a corresponding class label $Y_i \in \{0, 1, \cdots, C\text{-}1\}$, where C is the total number of classes in $\mathcal{D}$. Time series classification is to predict a label $\hat{Y}_i$ of X_i by a specific model.

Efficient transformer. A transformer-based model has a time complexity $O(N^2)$, where N is the number of input. The goal of an efficient transformer is to minimize the model's training time.

2.1 Self-Attention Mechanism

Most transformer-based methods apply the softmax self-attention mechanism in their models [47, 49, 63], which results in the quadratic time complexity to them. We briefly review the self-attention mechanism in the following.

In the vanilla self-attention mechanism, the input token S is a set of N embeddings and the embedding type depends on the specific tasks, *e.g.*, word embedding for NLP [9, 47], image patch for CV [10, 26], and time series subsequences for time series [29, 63]. The input S is transmitted to three projections: Q (query), K (key), and V (value), which are defined as follows:

$$Q = SW_q, K = SW_k, V = SW_v \tag{1}$$

where W_q, W_k, $W_v \in \mathbb{R}^{d \times d}$ are learnable weights and d is the dimension of input. The matrix calculation of the self-attention mechanism is defined as follows:

$$Attn(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d}})V \tag{2}$$

Let q_i, k_j be the i-th and j-th row of Q, K. For a query $q_i \in Q$, the Equation 2 can be formulated in a vector format:

$$Attn(q_i, K, V) = \sum_{j=1}^{N} \frac{\exp(q_i^\top k_j)}{\sum_{j'=1}^{N} \exp(q_i^\top k_{j'})} v_j^\top \tag{3}$$

Intuitively, Equation 2 determines the normalized similarity scores across all query-key pairs (q and k), which are then utilized to weigh the value vectors v. Consequently, the time complexity of softmax attention is $O(N^2 d)$. Typically, since $d \ll N$, the time complexity of the self-attention mechanism simplifies to $O(N^2)$.

2.2 Random Feature Attention

To tackle the quadratic time complexity in softmax self-attention, statistical RFA methods are proposed to build the self-attention based on kernels [5, 60] without any *priors*.

The core idea of RFA is to take the exponential function in Equation 3 as an exponential kernel function $\exp(x^\top y)$. Ideally, two high-dimension projections $\phi(x)$ and $\phi(y)$ of the input x and y, respectively, are employed to calculate the exponential kernel as follows:

$$\exp(x^\top y) = \phi(x)\phi(y) \tag{4}$$

In practice, we can hardly find the exact projection ϕ which satisfies Equation 4. Rahimi et al. [38] propose that ϕ can be approximated using $f(\omega^\top x)$ based on Monte Carlo method:

$$\phi(x) \approx \frac{h(x)}{\sqrt{m}} (f(\omega_1^\top x), ..., f(\omega_m^\top x), ..., f(\omega_1^\top x), ..., f(\omega_m^\top x))^\top, \tag{5}$$

where $\omega_1, \omega_2, ..., \omega_m$ are random features selected from a Gaussian distribution, $h(x)$, $f(x)$ are two given functions of a RFA method. Under the guidance of Bochner's theorem [2], Equation 4 can be reformulated as the following expectation [60]:

$$\exp\left(x^\top y\right) = \mathbb{E}_{\omega \sim \mathcal{N}(\omega; 0, \mathbf{I})} \left[\phi(x, \omega)^\top \phi(y, \omega)\right], \tag{6}$$

where $\mathcal{N}(\omega; 0, \mathbf{I})$ denotes a Gaussian distribution with the mean of zero and identical covariance matrix. The function $\phi(\cdot, \cdot)$ denotes a random projection from the input vector $x \in \mathbb{R}^d$ to a high dimension vector $\phi(x, \omega) \in \mathbb{R}^{2m}$ as shown in Equation 5. Here m is the number of random features sampled from a Gaussian distribution.

After approximating the expectation, q_i and k_j are designated as x and y in Equations 5 and 6, respectively. Subsequently, these variables are incorporated into the vector format of softmax self-attention in Equation 3 to derive the random feature attention (namely, *RFAttn*) as follows:

$$\begin{aligned} RFAttn\left(q_i, K, V\right) &= \sum_{j=1}^{N} \frac{\exp(q_i^\top k_j)}{\sum_{j'=1}^{N} \exp(q_i^\top k_{j'})} v_j^\top \\ &\approx \frac{\sum_{j=1}^{N} \phi\left(q_i, \omega\right)^\top \phi\left(k_j, \omega\right) v_j^\top}{\sum_{j'=1}^{N} \phi\left(q_i, \omega\right)^\top \phi\left(k_{j'}, \omega\right)} \\ &= \frac{\phi\left(q_i, \omega\right)^\top \sum_{j=1}^{N} \phi\left(k_j, \omega\right) v_j^\top}{\phi\left(q_i, \omega\right)^\top \sum_{j'=1}^{N} \phi\left(k_{j'}, \omega\right)} \end{aligned} \tag{7}$$

In Equation 7, the original exponential kernel between q_i and k_i is replaced by a series of functions $\phi(\cdot, \cdot)$. Thus, it only calculates $\sum_{j=1}^{N} \phi\left(k_j, \omega\right) v_j^\top$ and $\sum_{j'=1}^{N} \phi\left(k_{j'}, \omega\right)$, and uses the results for all q_i to compute the attention. Because there are N queries, the final time complexity of RFA is $O(N)$.

Problem statement. This paper investigates an efficient transformer method for time series, and applies it to solve TSC. □

Table 1: Summary of Frequently Used Notations

Notation	Description
X	a time series instance
T	a time series subsequence generated from X
s	a shape, which is d-dimension vector
S	an input for a transformer, consisting of N shapes
$\mathcal{D}$	a time series dataset, consists of $\mathbb{N}$ time series
$\mathcal{T}$	time series subsequences generated from $\mathcal{D}$
$\mathcal{S}$	a shape set for $\mathcal{D}$, consists of $\mathbb{N}$ shapes S
C	a cluster center
N	the number of input shapes (*i.e.*, the input sequence length)
W	the weights for training
$\phi_1,\ \phi_w, \phi_2$	the projection functions of input matrix
Φ_i	a set of projections $\phi_1,\ \phi_w, \phi_2$ for a time series instance X_i

3 CHALLENGES OF RFA ON TIME SERIES

In this section, we present a comprehensive analysis of the challenges faced by RFA methods in time series, specifically focusing on efficiency, overflow, and approximation error.

3.1 Efficiency

The precision of ϕ for approximating the exponential kernels is crucial for the RFA method. Several works [15, 35] tried to minimize the approximation error between RFA by optimizing ϕ. A classical choice [35, 38] is to let $h(x) = \exp\left(\frac{1}{2}\|x\|^2\right)$, and $f(\omega, x) = \left[\sin\left(\omega^\top x\right), \cos\left(\omega^\top x\right)\right]^\top$. By substituting them into the projection ϕ of Equation 5, we obtain Equation 8 as follows:

$$\phi^{trig}(x,\omega) = \frac{1}{\sqrt{m}} \exp\left(\frac{\|x\|^2}{2}\right)\left(\sin\left(\omega_1^\top x\right), \ldots, \sin\left(\omega_m^\top x\right), \cos\left(\omega_1^\top x\right), \ldots, \cos\left(\omega_m^\top x\right)\right)^\top. \tag{8}$$

Since the projection ϕ in Equation 8 uses trigonometric functions for random features, the corresponding attention is named as trigonometric random feature attention.

A positive random feature attention is further proposed in Performer [5], which reduces the approximation error of trigonometric attention. It makes $h(x) = \exp\left(-\frac{\|x\|^2}{2}\right)$ and $f(\omega, x) = \left[\exp\left(\omega^\top x\right), \exp\left(-\omega^\top x\right)\right]^\top$, then substitutes them into Equation 5. The projection ϕ^{pos} of positive random feature attention in Performer is shown as below:

$$\phi^{pos}(x,\omega) = \frac{1}{\sqrt{2m}} \exp\left(-\frac{\|x\|^2}{2}\right)\left(\exp\left(\omega_1^\top x\right), \ldots, \exp\left(\omega_m^\top x\right), \exp\left(-\omega_1^\top x\right), \ldots, \exp\left(-\omega_m^\top x\right)\right)^\top. \tag{9}$$

After applying ϕ^{pos} to each q_i and k_i, Q' and K' are expressed using $\phi^{pos}(q,\omega)$ and $\phi^{pos}(k,\omega)$. The matrix calculation of positive random attention is written as follows:

$$PosAttn(Q,K,V) = \widehat{D}^{-1}\left(Q'\left(\left(K'\right)^\top V\right)\right), \quad \widehat{D} = diag\left(Q'\left(\left(K'\right)^\top 1_L\right)\right). \tag{10}$$

For previous RFA-based methods, when calculating the projection $\phi(x,\omega)$ in Equation 8 and 9, the random features $\omega_1^\top, \omega_2^\top, \cdots, \omega_m^\top$ need to be assigned for all time series in each epoch during the training process. Thus, their efficiency is constrained by the number of random features m.

3.2 Overflow

Given a shape $s \in S$ identified from the shape set $\mathcal{S}$ in TSC datasets [1]. An overflow will occur when taking s as the input x for RFA methods because of calculating the term $\exp(\frac{\|x\|^2}{2})$ in Equation 8, 9. To better illustrate the overflow in RFA methods, an example is provided in Example 1:

Example 1. *Considering an input shape s from BasicMotions, $s = [10.31, -11.37, -13.42, -13.42, 13.88, 14.75, 16.54, 14.25, 5.05, -12.29]$, the length of shape s is 10, and the value of $\|s\|^2$ is 1659.45. Take s as the input x for Equation 8 or 9 to calculate random feature attention. The term $\exp(\frac{\|x\|^2}{2})$ exceeds the upper bound of Python 2^{1024}. As a result, it leads to the overflow for computing both trigonometric and positive random feature attention.*

In practice, the overflow can be found across many time series datasets shown in Section 5.3.2, leading to the failure of updating weights when training a transformer model on these datasets.

3.3 Approximation error

For a given RFA method, the approximation error of RFA to vanilla softmax attention is defined as follows:

Definition 1. *The approximation error for positive or trigonometric random feature attention to vanilla softmax attention is defined by Mean Square Error (MSE):*

$$\begin{aligned} MSE(RFAttn) &= \frac{1}{N}\sum_{i=1}^{N}\left(RFAtt(q_i,K,V) - SMAtt(q_i,K,V)\right)^2 \\ &= \frac{1}{N}\sum_{i=1}^{N}\Big[\sum_{j=1}^{N}\Big(\frac{\phi\left(q_i,\omega\right)^\top \phi\left(k_j,\omega\right)}{\sum_{j'=1}^{N}\phi\left(q_i,\omega\right)^\top \phi\left(k_{j'},\omega\right)} \\ &\quad - \frac{\exp(q_i^\top k_j)}{\sum_{j'=1}^{N}\exp(q_i^\top k_{j'})}\Big)v_j^\top\Big]^2 \end{aligned} \tag{11}$$

Drawing from Definition 1, Lemma 3.1 is established as follows:

Lemma 3.1. *Given a trigonometric or positive random feature attention, $\mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left[RFAttn(q_i,K,V)\right] \neq SMAttn(q_i,K,V)$ based on Equation 6.*

Proof. Make $a_{ij} := \phi(q_i,\omega)^\top\phi(k_j,\omega)$. With Equation 6 and the property of expectation, we achieve $\mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left[\sum_{j=1}^{N} a_{ij}\right] = \sum_{j=1}^{N}\exp\left(q_i^\top k_j\right)$. However, a_{ij} and $\frac{1}{\sum_{j=1}^{N} a_{ij}}$ are generally not independent, *i.e.*, $Cov(a_{ij}, \frac{1}{\sum_{j=1}^{N} a_{ij}}) \neq 0$.

Thus we have $\mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left(\frac{a_{ij}}{\sum_{j=1}^{N} a_{ij}}\right) \neq \mathbb{E}(a_{ij})\cdot\mathbb{E}\left(\frac{1}{\sum_{j=1}^{N} a_{ij}}\right)$. Based on the above equations, we can rewrite it as:

$$\mathbb{E}_{\omega\sim\mathcal{N}(\omega;0,\mathbf{I})}\left(\frac{a_{ij}v_j^\top}{\sum_{j=1}^{N} a_{ij}}\right) \neq \frac{\exp\left(q_i^\top k_j\right)v_j^\top}{\sum_{j=1}^{N}\exp\left(q_i^\top k_j\right)} \tag{12}$$

□

The approximation error, as detailed in Definition 1, does exist in practice and influences the accuracy of classification of a transformer model. In addition, from Lemma 3.1, while $\phi(q_i,\omega)^\top\phi(k_j,\omega)$ in Equation 6 is an unbiased estimation of $\exp(q_i^\top k_j)$, $Q'(K')^\top$ in Equation 10 still holds a biased estimation to $\text{softmax}(QK^\top)$. This produces a high approximation error, as we have shown in Figure 1. Consequently, the classifiers based on RFA methods can exhibit unstable accuracy results.

Discussion. In summary, we identify the fundamental issue as the reliance of RFA methods on a single fixed projection (Equations 8, 9) to mathematically approximate the softmax kernel, without considering the original data. Importantly, the distribution of time series data varies significantly across different real-world applications. For example, the time series in human activity recognition exhibits a larger data range and variance compared to the time series in EMA (ElectroMagnetic Articulograph), which records the movement of the human's lips and tongue [1]. Therefore, it is desirable to have a data driven approach on ϕ when handling time series.

4 EFFICIENT TRANSFORMER – DARKER

In this section, we propose an efficient transformer, called DARKER, for time series. In particular, we first introduce the data-driven kernel-based attention mechanism, which utilizes the projections learned from multiple time series in Section 4.1. We then construct an index, called pIndex to efficiently search the projection fitting S in Section 4.2. In Section 4.3, we optimize shape discovery from time series subsequences via an index named I_{c2d} after clustering. Finally, Section 4.4 discusses several optimizations in DARKER.

4.1 DARKER Attention Mechanism

To tackle time series data, we take the number of shapes N as an input for our transformer model with DARKER.[2] Extracting shapes is treated as a preprocessing in our approach.

We propose to use machine learning models, with their learnable weights, as a data-driven projection for input data to approximate the softmax kernel in DARKER. These models trained on input data demonstrate superior data fitting capabilities and avoid computing the $\exp(\frac{\|x\|^2}{2})$ function in both trigonometric (Equation 8) and positive (Equation 9) random feature attention methods, effectively mitigating the overflow.

Our solution has two stages, namely learning projections and data-driven attention.

4.1.1 DARKER *Learning projections.* We begin with Equation 1, which describes the query-key-value mapping to input before attention mechanism. This is subsequently incorporated into Equation 2, as details below:

$$\begin{aligned} Attn(Q,K,V) &= \text{softmax}(QK^\top)V \\ &= \text{softmax}(SW_Q(SW_K)^\top)V \\ &= \text{softmax}(S(W_QW_K^\top)S^\top)V \\ &= \text{softmax}(SW_SS^\top)V \end{aligned} \tag{13}$$

where $W_S = W_QW_K^\top$ represents the learnable weights for the attention mechanism. The matrices Q and K with a size of $N * d$ in Equation 2 are transformed into two fixed matrices $S, S^\top$ and a size of $d * d$ learnable weight matrix W_S. S and $S^\top$ are the input shape and its transpose.

Our Equation 13 transforms the uncertain matrices Q and K into two fixed matrices $S, S^\top$, and a small size changing $d * d$ matrix, W_S. For a specific time series instance, the operation allows us to build a model on $S, S^\top$ with a changing W_S, instead of on Q and K. Considering that the size of W_S $(d * d)$ is much smaller than that of Q and K $(N * d)$, it becomes easier to train a model as the projection for S, W_S, and $S^\top$.

The task then launchs into finding the projections for the three matrices S,W_S, and $S^\top$ that satisfy the following Equation:

$$\begin{aligned} Attn(Q,K,V) &= \text{softmax}(SW_SS^\top)V \\ &\approx \phi_1(S)\phi_w(W_S)\phi_2(S^\top)V \end{aligned} \tag{14}$$

Since the data distribution of time series varies, even time series from the same dataset can be significantly different. To ensure that the model can adapt to different time series instances, we separately train the model for each time series and build an index for storing the model instead of training a single one for all time series. The weights for all input shapes are obtained during the model training phase, and then an index is created to query the weights of the model for a specific time series. We denote the projection for the i-th shape S_i as Φ_i.

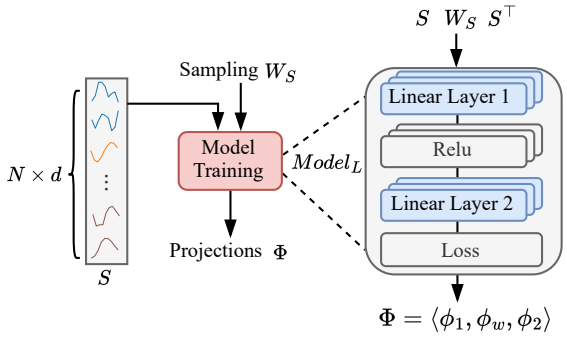


Figure 3: Learning projections in DARKER

It is critical to note that MLPs can approximate arbitrary continuous functions [14, 22, 23]. Thus we design a neural network consisting of two linear layers, namely $Linear_1$ and $Linear_2$, and a ReLU layer as the basic block, which are collectively referred to as a linear block, for our model. We build our model $model_L$ by applying three linear blocks to the three inputs as shown in the right part of Figure 3. As a result, each of S, W_S, $S^\top$ has its corresponding linear block. We denote these three linear blocks as $\phi_1()$, $\phi_w()$, $\phi_2()$, and make $\Phi = \langle \phi_1, \phi_w, \phi_2 \rangle$. Formally, we have Equation 15 following Equation 13:

$$\begin{aligned} \text{softmax}(SW_SS^\top) &\approx model_L(S, W_S, S^\top) \\ &= Linear_2(Relu(Linear_1(S)))* \\ &\quad Linear_2(Relu(Linear_1(W_S)))* \\ &\quad Linear_2(Relu(Linear_1(S^\top))) \\ &= \phi_1(S)\phi_w(W_S)\phi_2(S^\top) \end{aligned} \tag{15}$$

Here, S, W_S, and $S^\top$ are the inputs of our $model_L$, where $Linear_1$ and $Linear_2$ within the ReLU activation function are used to approximate the corresponding kernel function of W_S. Moreover, the weight matrix W_S is shared among the shapes in input S.

Example 2. *Assume that we have* $S_1 = [s_1, s_2, s_3, s_4]^\top$, *where* $s_1 = [0.78, 0.25]$, $s_2 = [0.51, 0.93]$, $s_3 = [0.18, 0.68]$, *and* $s_4 = [0.77, 0.87]$. *After applying them to* $model_L$, *we have corresponding projections* ϕ_1, ϕ_w, *and* ϕ_2. *Based on Figure 3 and Equation 15, we obtain the hypothetical weight matrices for the projection* ϕ_1 *in linear layer 1* $[0.66, 0.23, 0.98, 0.87]$ *and linear layer 2* $[0.54, 0.12, 0.78, 0.98]^\top$. *Similarly, we also have those corresponding matrices for* ϕ_w $(1 : [0.57, 0.34, 0.87, 0.23], 2 : [0.91, 0.78, 0.26, 0.46]^\top)$ *and* ϕ_2 $(1 : [0.12, 0.56, 0.90, 0.34], 2 : [0.98, 0.54, 0.21, 0.87]^\top)$.

We use Mean Square Error (MSE) loss as presented in Equation 16 because the purpose of the model is to approximate the softmax kernel.

$$\mathcal{L}_{MSE} = \|\text{softmax}(SW_SS^\top) - \phi_1(S)\phi_w(W_S)\phi_2(S^\top)\|^2 \tag{16}$$

The details of learning projections for approximating softmax kernel are presented in Algorithm 1. We define $\mathcal{S}_\Phi$ as the input shape set where projections of shapes are learned while $\overline{\mathcal{S}}_\Phi$ =

[2]The effectiveness of the discriminative time series subsequences (simply called shapes) for TSC has been demonstrated [13, 19, 21].

Algorithm 1: DARKER's learning projections

Input: Shape set $\mathcal{S}_\Phi$, corresponding cluster centers C, the number of training epochs *Epochs*
Output: Index I

1 I.init(); // Initialize I as a hashmap
2 **for** S *in* $\mathcal{S}$ **do**
3 Initialize a model $model_L$;
4 **for** $epoch \in \{0, \cdots, epochs\}$ **do**
5 // Using $model_L$ to approximate softmax
6 $loss \leftarrow \texttt{MSE}(\text{softmax}(SW_S S^\top) - model_L(S, W_S, S^\top))$;
7 $model_L$.backward($loss$);
8 $\langle \phi_1, \phi_w, \phi_2 \rangle \leftarrow model_L.weights$;
9 $\Phi \leftarrow \langle \phi_1, \phi_w, \phi_2 \rangle$;
10 I.insert(S, Φ);
11 **return** I

$\mathcal{S} - \mathcal{S}_\Phi$ as the shape set where projections of the shapes have not been learned. We first initialize an empty hashmap as index I (Line 1). For each S in $\mathcal{S}_\Phi$ (Line 2), we initialize $model_L$ (Line 3). We then train our $model_L$ on S with the loss function in Equation 16 (Lines 4-7). The weights of $model_L$ with corresponding S are stored in a hashmap I (Lines 8-10). This hashmap is further extended to a PINDEX for searching the projections of $S_\theta \notin \mathcal{S}_\Phi$ in Section 4.2.

4.1.2 DARKER *Data-driven attention.* After learning projections, we construct a hashmap I, which is used in data-driven attention for querying projections to input. We illustrate our data-driven attention using the hashmap I (Figure 4(b)) and vanilla attention mechanism (Figure 4(a)), where $(\cdot)$ means the multiplication of two matrices.

In our data-driven attention, we first query S in hashmap I to obtain its corresponding Φ. Since all S in $\mathcal{S}_\Phi$ have been stored in I, the time complexity of searching the trained weights Φ for $S \in \mathcal{S}_\Phi$ is $O(1)$. The obtained Φ_i is then used to compute the projections of three matrix S, W_S, $S^\top$, as shown in the following Equation 17:

$$\begin{aligned} DARKERAttn(Q,K,V) &= \Phi(Q,K)V \\ &= \phi_1(S)\phi_w(W_Q W_K)\phi_2(S^\top)V \\ &= \phi_1(S)(\phi_w(W_Q W_K)(\phi_2(S^\top)V)) \end{aligned} \tag{17}$$

Due to the associative property of matrix multiplication, we first multiply $\phi_2(S^\top)$ and V to obtain a matrix with dimension $d * d$ in Equation 17. We then use the obtained result to multiply matrix $\phi_w(W_Q W_K)$ and matrix $\phi_1(S)$. Under this reverse order, we avoid the computation to a large $N*N$ matrix, reducing the computational complexity to $O(Nd)$.

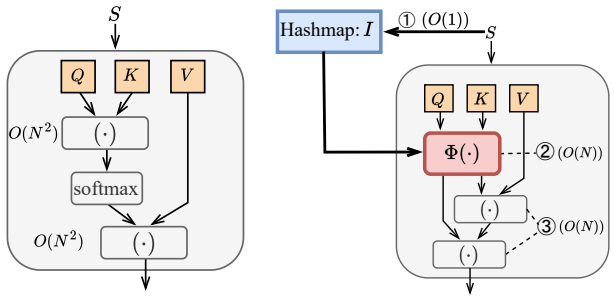
The steps of training a transformer model (named $model_T$) for TSC are shown in Algorithm 2. $model_T$ is training on all input S_i for many epochs (lines 2-3). It is worth noting that the query operation (line 4) only needs to be executed once for all training epochs with respect to an input. Then, after obtaining the loss (line 5) and using it for backward propagation (line 6), the output $model_T$ is used in inference for predicting the class label.

In a nutshell, DARKER differentiates itself from existing RFA methods by training a $model_L$ for fitting input data when learning projections and utilizing $model_L$ to train $model_T$ for downstream tasks. As opposed to manually selecting a fixed projection in other methods, this approach aims to approximate the softmax kernel with projections learned from input, which improves the efficiency of transformer models and tackles the challenges in Section 3.

Algorithm 2: Training of transformer based on DARKER

Input: Shape set $\mathcal{S}_\Phi$, Index I, label Y, the number of training epochs *epochs*
Output: trained transformer model $model_T$

1 Initialize a transformer classification model $model_T$ with DARKER;
2 **for** *epoch in epochs* **do**
3 **for** S *in* $\mathcal{S}_\Phi$ **do**
4 $\Phi \leftarrow I.get(S)$;// $\Phi = \langle \phi_1, \phi_w, \phi_2 \rangle$
5 $loss \leftarrow model_T(S, \Phi, Y)$;
6 $model_T$.backward($loss$);
7 **return** $model_T$



(a) Vanilla attention (b) DARKER

Figure 4: Vanilla and DARKER attention mechanism

4.1.3 DARKER *Time complexity analysis.* We analyze and provide the time complexity of DARKER. As shown in Figure 4(b), we have: ① Query time (in a hashmap): $O(1)$. ② The time complexity of Φ: The time complexity of three linear blocks in Φ are $O(Nd)$, $O(d^2)$, and $O(Nd)$, respectively. ③ The matrix multiplication time: The time complexity of two matrix multiplications is both $O(Nd)$. Therefore, the total time complexity of the DARKER attention mechanism is $O(1 + Nd + Nd) = O(N)$ since $d \ll N$.

4.2 Search Projections for $\overline{\mathcal{S}}_\Phi$

In Section 4.1.1, we discussed the step of learning projections and the use of an index I implemented as a hashmap. However, in practical applications, users are highly likely to encounter an input $S_\theta \in \overline{\mathcal{S}}_\Phi$ whose projection has not been learned and stored in I, and DARKER should be capable of handling them. To tackle this problem, we propose a PINDEX for efficiently searching projection for both $S_\theta \in \overline{\mathcal{S}}_\Phi$ and $S_i \in \mathcal{S}_\Phi$. Since we have introduced the process of directly using a hashmap for S_i, we present the details of searching projection Φ_θ for S_θ as follows.

First, we show the precondition of searching projection for S_θ in Lemma 4.1:

Lemma 4.1. *Suppose* $S_\theta^{N*d} \in \overline{\mathcal{S}}_\Phi$ *and* $S_i^{N*d} \in \mathcal{S}_\theta$. $\forall\ S_i$ *and* S_Φ,

$$\begin{aligned} &\Pr[\|S_\theta - S_i\| \le \varepsilon_0] = 1 - O(p^{Nd}) \\ &\Rightarrow \Pr[\|S_\theta W_S S_\theta^\top - S_i W_S S_i{}^\top\| < \varepsilon] \ge 1 - O(p^{Nd}) \end{aligned} \tag{18}$$

where $\varepsilon_0 = o(1)$, $\sigma_i = \|S_\theta - S_i\|$, *and* $p = \frac{\sigma_i}{\max(\sigma_i)}$, ε *is a variable with quadratic dependency on* σ_i.

The proof of Lemma 4.1 is given in Appendix A.5.1. In Lemma 4.1, Equation 18 provides a rough bound for the accuracy of our approximation on $S_\theta W_S S_\theta^\top$ using $S_i W_S S_i{}^\top$. Here $\sigma_i = \|S_\theta - S_i\|$ denotes the distance between S_θ and S_i, $\varepsilon_0 = o(1)$ represents a infinitesimal and $p = \frac{\sigma_i}{\max(\sigma_i)}$ is a ratio of given σ_i to the maximum σ when S_i changes. The other ε denotes a quadratic polynomial with respect to σ_i, and its specific form is shown in the proof of Lemma 4.1.

Consequently, large $|\mathcal{S}_\Phi|$ monotonically results in a discrete distribution of S_i, possibly leading to a decrease in p. Thus we can reduce the potential for arbitrary errors by replacing Φ_θ with Φ_i. Note that the performance can be enhanced by increasing the size of $\mathcal{S}_\Phi$ when we sample finite representative shapes. This can also be confirmed by our experiments in Section 5.4.2.

We then show the correctness of replacing Φ_θ with Φ_i as following Theorem 4.2:

Theorem 4.2. *Suppose* $S_\theta^{N*d} \in \overline{\mathcal{S}}_\Phi$ *and* $S_i^{N*d} \in \mathcal{S}_\theta$. $\forall\ S_i$ *and* S_Φ,

$$\begin{aligned} &\Pr[\|\Phi_\theta(S_\theta W_S S_\theta^\top) - \Phi_i(S_\theta W_S S_\theta^\top)\| < (L_\theta + L_i)\varepsilon + 2\eta_i] \\ &\ge [1 - O(p^{Nd})]^2 \cdot [1 - O(q^{Nd})]^{N-1} \end{aligned} \tag{19}$$

where L_θ *and* L_i *are two positive constants,* ε *is a variable with quadratic dependency of* σ_i, $\sigma_i = \|S_\theta - S_i\|$, $\eta_i = \|\Phi_i(S_i W_S S_i^\top) - \text{softmax}(S_i W_S S_i^\top)\|$, $p = \frac{\sigma_i}{\max(\sigma_i)}$, *and* $q = \frac{\eta_i}{\max(\eta_i)}$.

The proof of Theorem 4.2 can be found in Appendix A.5.2. Equation 19 gives us another rough bound about the approximation of projections based on Equation 18. In addition to the existing symbols in Lemma 4.1, L_θ and L_i are two Lipsitz constants, η_i represents the difference between $\Phi_i(S_i W_S S_i^\top)$ and $\text{softmax}(S_i W_S S_i^\top)$, and q denotes the ratio of η_i to the maximum η when Φ_i changes.

In Theorem 4.2, a smaller $\sigma_i = \|S_\theta - S_i\|$ and $\eta_i = \|\Phi_i(x_i) - SM(x_i)\|$ indicates larger probability of finding existing Φ_i to accurately approximate Φ_θ. Therefore, to find projection Φ_θ fitting S_θ, we need to search for the nearest S_i in the shape set $\mathcal{S}_\Phi$ to S_θ. Lemma 4.1 and Theorem 4.2 illustrate that Φ_θ approximately equals to Φ_i on the same domain field $S_\theta W_S S_\theta^\top$.

In such case, our problem can finally be reduced to an Approximate Nearest Neighbor (ANN) search task on $\mathcal{S}$. Each element from $\mathcal{S}$ is a matrix S with size $N \times d$, which means we need to search a matrix (two dimensions) rather than a vector (one dimension) in most existing ANN methods [20, 36]. This two-dimensional distinction leads to the inefficient search process.

Our proposed solution, pIndex, leverages the clustering process during shape discovery as details in Section 4.3. This process serves as a method of feature reduction to achieve a low-dimension representation R for S. Specifically, we take C_S, which represents the clusters of all shapes within S identified in shape discovery, as the representation R. We justify the use of representation R in pIndex as follows: each shape s, one row of S, is the nearest time series subsequence to the cluster center and is selected in the order of these cluster centers. If most shapes from S_i and S_θ in the same row belong to the same cluster, this suggests a smaller $\sigma_i = \|S_\theta - S_i\|$ in Theorem 4.2. Therefore, it is more likely that S_i and S_θ share the same Φ.

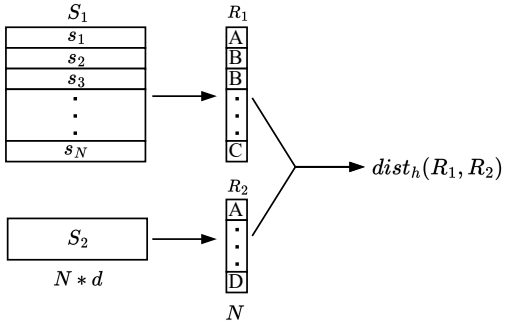
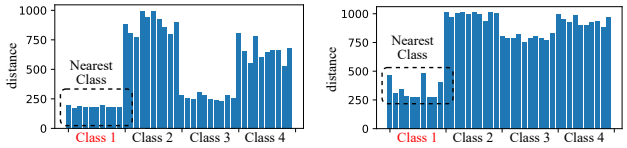


Figure 5: The process of transmitting S to representation R and computing the Hamming Distance



(a) Euclidean Distance Based on S (b) Hamming Distance Based on R

Figure 6: The distance of a selected instance from Class 1 to others

Given that R represents the clusters to which the shapes in S belong, we can consider R as a sequence composed of N symbols, with each symbol corresponding to a specific cluster. To quantify the similarity between two such representations, R_1 and R_2, we employ the Hamming Distance metric focusing on their respective cluster centers. The definition of the Hamming Distance is provided in the Equation 20:

$$d_H(R_1, R_2) = \sum_{i=1}^{N} \delta(r_1^i, r_2^i) \tag{20}$$

where r_1^i, r_2^i are the i-th value of two vector R_1, R_2, and $\delta(r_1^i, r_2^i)$ denotes Kronecker Function measuring if r_1^i and r_2^i are equal.

Figure 5 illustrates the process of utilizing clusters as representations and calculating Hamming distances based on these representations. An example of this process is provided below:

Example 3. *Suppose* S_1 *and* S_2. *Each of them consists of four shapes,* $S_i = [s_1, s_2, s_3, s_4]$, *where* s *is a vector with length* d. *The cluster centers of* s *in* S_i *are* C_i. *We use the cluster centers as the representation* $R_1 = [A, A, B, C]$ *for* S_1 *and* $R_2 = [A, B, B, B]$ *for* S_2. *Thus, the Hamming Distance between* R_1 *and* R_2 *is* 2.

To demonstrate the effectiveness of our proposed representation R, we randomly select a time series instance from Class 1 in the test set. We compute the Hamming Distance based on R and the Euclidean Distance based on S against all instances in the training set. The results are shown in Figure 6, where the x-axis represents the IDs of the time series in the training set, divided by class label. Our observations indicate that the nearest class to the randomly selected time series instance, for both distance measures, is Class

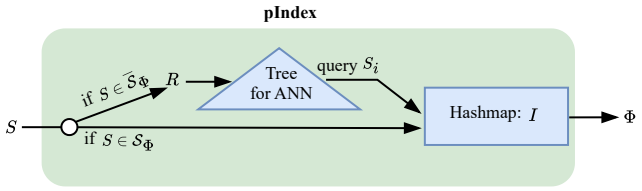


Figure 7: pIndex of searching Φ for S

1—consistent with the instance's true class label. Furthermore, we notice that the Hamming distances for time series from Class 1 to the other classes are significantly larger compared to the distances within Class 1. In contrast, the Euclidean distances exhibit minimal disparity between Class 1 and Class 3. These results underscore the superiority of representation R and the effectiveness of employing Hamming Distance over Euclidean Distance in this context. Moreover, the dimension reduction from S to R coupled with the application of Hamming Distance significantly enhances the efficiency of ANN within our pIndex.

We apply the Ball Tree based on Hamming Distance to searching the nearest neighbor for S in our pIndex, because of its efficiency in handling high-dimensional data [32]. We note that there may be other solutions for ANN. As the selection or proposal of a better index for ANN is orthogonal to our method, we simply use Ball Tree here. Any other advanced index for ANN to data series based on Hamming Distance can further improve our efficiency.

Finally, we show the details of pIndex in Figure 7. When the input S is in $\mathcal{S}_\Phi$, we directly use a hashmap from Section 4.1.1 to obtain its projection; when S is in $\overline{\mathcal{S}}_\Phi$, we conduct a ANN search to obtain $S_i \in \mathcal{S}_\Phi$ based on the representation R of S and then take the projection *Phi* from a hashmap. We also give an example for illustrating pIndex as follows:

Example 4. *Assume there is a $S_\theta = [s_1, s_2, s_3, s_4]^\top$, the cluster centers of them are $[A, A, B, C]$ after clustering. So we have its representation $R_\theta = [A, A, B, C]$. We query R_θ in a ball tree based on Hamming Distance to find the nearest $R_i = [A, A, B, B]$ to R_θ. Then, the projection Φ_i of S_i is taken as Φ_θ for S_θ.*

It should be remarked that each input is required to search for its projection in pIndex only once at the beginning of the training and then used in all epochs of training. In our datasets, the height of a ball tree in pIndex is always smaller than a constant 32, which is much smaller than the number of epochs. Hence, the training time is dominated by matrix multiplications in each epoch, presented in Section 4.1.3.

4.3 Optimized Preprocessing of Shape Discovery

[13, 19, 21, 63] have confirmed that the discriminative time seires subsequences (or simply called shapes) lead to a better performance on tsctasks. Inspired by the previous works, we also employ a preprocessing to discover shapes based on clustering algorithm (*e.g.*, kmeans) and take them as the input for a transformer model. However, the discovery process is time-consuming. Thus, we propose an efficient shape discovery algorithm shown in Algorithm 3.

The details of selecting time series subsequences as shapes for representing time series can be described as follows. First, we initialize a shape set $\mathcal{S}$ and a set $\mathcal{T}$ of time series subsequences (Line 1).

Algorithm 3: Shape discovery for DARKER's input

Input: Time series dataset $\mathcal{D}$
Output: shape set $\mathcal{S}$
1 Initialize set $\mathcal{S}, \mathcal{T}$;
2 **for** X_j *in* $\mathcal{D}$ **do**
3 $\quad \mathcal{T} \leftarrow \mathcal{T} \bigcup \texttt{slideTS}(X_j)$;
4 `// build` I_{c2d} `to store clusters and distance`
5 $\langle I_{c2d}, \{C_1, C_2, ..., C_K\}\rangle \leftarrow \texttt{kmeans}(\mathcal{T})$;
6 **for** X_j *in* $\mathcal{D}$ **do**
7 $\quad$ Initialize $S_j \leftarrow []$;
8 $\quad$ **for** C_i *in* $\{C_1, C_2, ..., C_N\}$ **do**
9 $\quad\quad$ `// retrieve the distance from` I_{c2d}
10 $\quad\quad \texttt{dist}(C_i, X_j) \leftarrow I_{c2d}.\texttt{get}(C_i, X_j)$;
11 $\quad\quad$ `// Select the nearest` T `to` C_i `as` s
12 $\quad\quad s_i \leftarrow \texttt{select\_min}(\texttt{dist}(C_i, X_j))$;
13 $\quad\quad S_j[i] \leftarrow s_i$;
14 $\quad \mathcal{S}.\texttt{add}(S_j)$;
15 **return** $\mathcal{S}$

Then, we apply sliding windows to all time series X_j from dataset $\mathcal{D}$ to generate a large number of time series subsequences as $\mathcal{T}$ (Line 2-3). Afterward, a kmeans is applied to these time series subsequences $\mathcal{T}$. As a result, we obtain their clusters $\{C_1, C_2, ..., C_N\}$ and build an index I_{c2d} based on the clusters and their distances to time series subsequences (Line 4-5). We define $\texttt{dist}(C_i, X_j)$ as a set of distances $\{\texttt{dist}(C_i, T_j^1), (C_i, T_j^2)...\}$ between the center of cluster C_i and time series subsequences $\{T_j^1, T_j^2, ...\}$, where $\{T_j^1, T_j^2, ...\}$ are derived from sliding over X_j. Given a X_j (Line 6), we retrieve the set of distances $\texttt{dist}(C_i, X_j)$ for a cluster C_i (Lines 7-10) and select the nearest time series subsequence T as a shape s_i to the i-th row of S_j (Line 11-13). We give a concise example to illustrate this process.

Example 5. *Assume that we have a time series instance $X = [0.21, 0.58, 0.34, 0.26]$. We apply a sliding window to the instance and obtain three time series subsequences $\{T^1, T^2, T^3\}$ from X, where $T^1 = [0.21, 0.58]$, $T^2 = [0.58, 0.34]$, and $T^3 = [0.34, 0.26]$. After running the* kmeans *algorithm, we have three cluster centers $\{C_1, C_2, C_3\}$, and $I_{c2d} = \{C_1 : 0.12, 0.32, 0.16; C_2...; C_3...\}$ for X, where 0.12, 0.32, and 0.16 are the distances from cluster center C_1 to T_1,T_2, and T_3, respectively. Since 0.12 is the minimum among the three distances, we take T_1 corresponding to cluster center C_1 as a shape s_1. After repeating the selection of the nearest time series subsequences to three cluster centers $\{C_1, C_2, C_3\}$, we obtain $S = \{s_1, s_2, s_3\}$, where s_1, s_2, s_3 are the shapes for X.*

Compared to existing methods [19, 63] that also use the nearest time series subsequence as the shape, we introduce I_{c2d} in DARKER to index distances from cluster centers to shapes during clustering, thereby avoiding recalculating distances by retrieving them from I_{c2d}. The process of extracting shapes S_j from a time series X_j as input for DARKER by querying (C_i, X_j) in I_{c2d} is shown in Figure 8.

Given a time series instance X_j from a dataset $\mathcal{D}$, we have $\texttt{slideTS}(X_j) = \{T_j^1, T_j^2, ..., T_j^t\}$. Considering clusters $\{C_i\}_{i=1}^N$ of $\mathcal{T} = \bigcup_{j=1}^{\mathbb{N}} \bigcup_{k=1}^{t} T_m^k$, a shape s_i for time series X_j corresponding

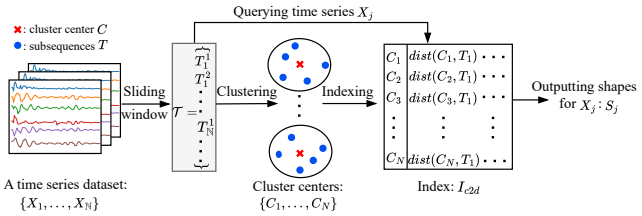


Figure 8: Extracting a shape for input X_j in cluster C_i

to a cluster C_i satisfies $dist(s_i, C_i) = min\{dist(T_m^k, C_i)\}, T_j^k \in X_j$. Proposition 4.3 justifies why we select the nearest time series subsequence as the shape for DARKER rather a random sample.

Proposition 4.3. *Considering random picking a s_i', it introduces more noise to the training data compared with the shape s_i, i.e., $dist(s_i', C_i) = dist(s_i, C_i) + \eta_i$ where η_i is the noise. Moreover, the variance of training data with s_i' $Var(s_i) = \frac{1}{\mathbb{N}-1} \sum_{i=1}^{\mathbb{N}} (s_i - C_i)^2$ increases with the noise η_i.*

After conducting Algorithm 3, every time series X is associated with a S, which contains N shapes. S is taken as an input to a transformer model. With N shapes in S, there exists a $O(N^2)$ time complexity efficiency problem, which is solved in Section 4.1. In addition, we store the clusters C_S of those shapes in S, which are used to obtain the representation R for S in Section 4.2.

4.4 Optimization and Implementation

For self-contained presentation, we present some techniques used for optimization and implementation of our method in DARKER and shape discovery with I_{c2d}.

For time series classification, the lengths of discriminative time series subsequences may vary [4, 39]. Therefore, in Section 4.3, in order to extract multi-scale shapes from time series, we follow the same strategy [19, 63] as existing works, which employs multi-scale windows, instead of a single length as the size of sliding windows. For all training processes of both $model_L$ in Algorithm 1 and $model_T$ in Algorithm 2, we employ the popular Adam optimizer, which has adaptive learning rates that adjust based on the gradient [16]. It allows the models to converge faster and handle various large, complex time series datasets. Since the model architecture is fixed as shown in Figure 3, storing $model_L$ as Φ for each time series in index is is equivalent to storing weights of the $model_L$. We only store the weights of $model_L$ for each time series in practice. To further prevent overfitting during training, we incorporate a dropout layer into our transformer model [41].

5 EXPERIMENTAL EVALUATION

In this section, we present an experimental evaluation DARKER. In particular, we report the experiment setup in Section 5.1. We then briefly introduce the compared benchmarked methods in Section 5.2. Section 5.3 reports the overall evaluation of DARKER, which includes both the efficiency (Section 5.3.1) and accuracy (Section 5.3.2). In Section 5.4, we conduct an experiment to show the efficiency of our pINDEX and evaluate the performance on various ratio for learning projections. Finally, we report the running time of our shape discovery and approximating error relative to vanilla attention in Section 5.5 and Section 5.6.

5.1 Experiment Setup

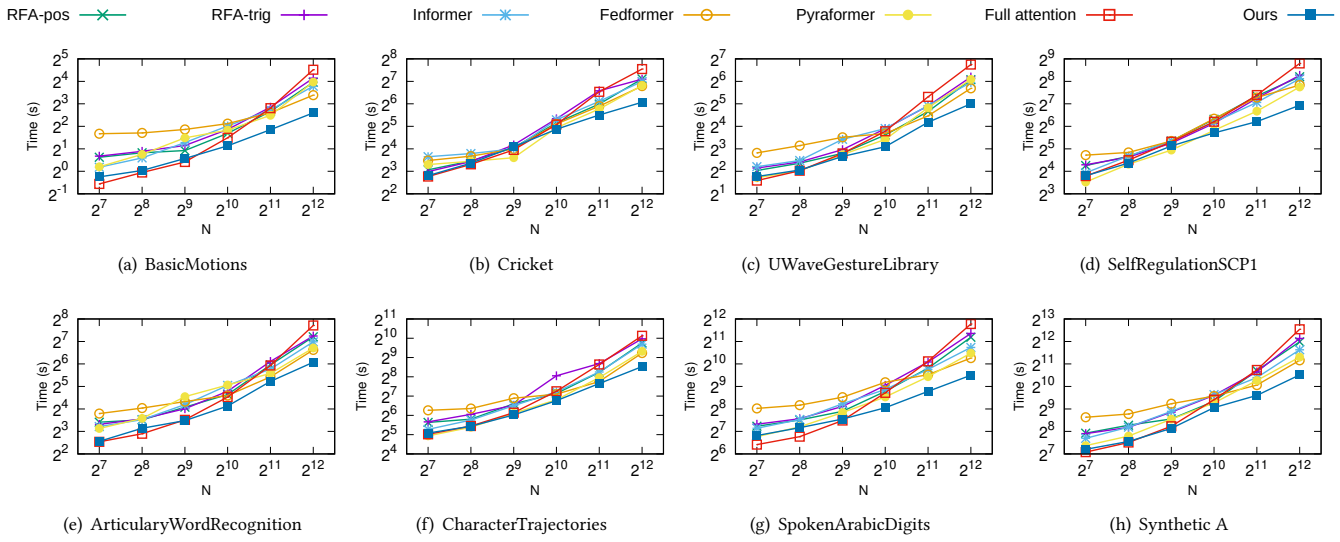
Environment. All experiments are conducted on the hardware of a machine with a single NVIDIA V100S GPU and one Xeon Gold 6226 CPU @ 2.70GHz. We use Python 3.8 and Pytorch 1.10.0 as the software environment. **Metrics.** We evaluate DARKER on both efficiency and effectiveness. For efficiency, we employ training/running times as evaluation metrics. Furthermore, we use FLOPs (floating-point operations per second) for the transformer model, which are commonly applied to measure the computational performance of a neural network model. For effectiveness, we utilize the accuracy metric, given that we present our work with TSC. In addition, following existing works [19, 58, 63], we calculate the average accuracy and average rank, and conduct Friedman and Wilcoxon signed-rank tests on all methods. **Datasets.** We use 14 UEA archive datasets [1] from various domains for TSC, supplemented by 2 synthetic datasets to assess performance on larger scales (with training sizes of 10,000 and 20,000). Some characteristics of the datasets are presented in Appendix A.1. **Implementation details.** For the experiment on training time, we set hyperparameters to be the same across all datasets. For training, we follow the previous work [58, 63] to split the training set into two parts, namely 80% and 20%, where the 20% part is as the validation set to tune the hyperparameters. The details of hyperparameters are also provided in Appendix A.3.

5.2 Benchmarked Methods

We compare DARKER against six benchmarked methods: the vanilla transformer [47] and five SOTA efficient transformers. Among the five efficient transformers, RFA-trig, RFA-pos employ random feature attention, while the remaining three are efficient transformers optimized for time series. **Full attention [47]:** Vanilla transformer models employ softmax attention (also called full attention) across all query-key pairs, resulting in $O(N^2)$ time complexity. **Informer [61]:** Informer, the SOTA efficient transformer for time series, introduces the ProbSparse attention mechanism, selectively processing dominant query-key pairs to achieve an improved $O(NlogN)$ time complexity. **Fedformer [62]:** Fedformer is a frequency-enhanced decomposed transformer for time series, which selects a fixed number of Fourier components and achieves linear computational complexity. **Pyraformer [24]:** Pyraformer applies a pyramidal attention module in the transformer for time series. The time complexity scales linearly with input length. **RFA-trig [35]:** An representative method of RFA mechanism uses trigonometric random features defined in Equation 8 to approximate attention. **RFA-pos [5]:** An advanced RFA approach within the Performer framework utilizes positive random features for efficient attention approximation (defined in Equation 9, 10). Besides, we compared our method with two non-transformer methods in terms of accuracy. **EDI [1]**: The 1-Nearest Neighbour classifier with Euclidean distance. **Rocket [8]:** The SOTA non-transformer method for TSCon UEA archive reported by the survey [39].

5.3 Overall Evaluation

5.3.1 Experiment on efficiency. We evaluate the efficiency of our method by reporting the training time of both 14 real and 2 synthetic datasets ordered by $\mathbb{N}$, which is the dataset size. We vary the number of shapes N (the number of inputs) as the x-axis and present their



(a) BasicMotions (b) Cricket (c) UWaveGestureLibrary (d) SelfRegulationSCP1

(e) ArticularyWordRecognition (f) CharacterTrajectories (g) SpokenArabicDigits (h) Synthetic A

Figure 9: Training time on varying the number of shapes N

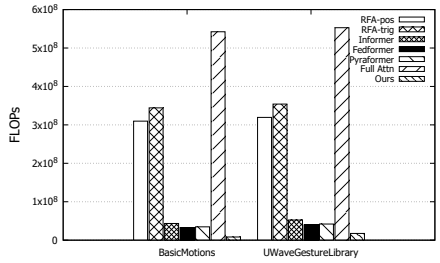


Figure 10: FLOPs of all methods on two datasets

training time as the y-axis. Due to space limitations, we report only the results for 8 out of 16 datasets in Figure 9, with consistent trends across all datasets. Full details are in the Appendix A.2.

In our experiments, DARKER demonstrates a clear improvement in computational efficiency. Specifically, it achieves a 3×-4× faster compared to full attention and 1.5×-3× improvement over other SOTAs at $N = 2^{12}$. Notably, as N grows from 2^7 to 2^{12}, our method's training time scales modestly by 6×-10×, contrasting with the full attention mechanism's 30×-44× increase, highlighting DARKER's scalability. Particularly on large datasets, DARKER shows substantial efficiency advantages, such as 4.8× faster processing on SpokenArabicDigits and 4× on Synthetic A, compared to full attention. This efficiency is consistent across dataset sizes, with a training time increase of less than 7-fold from 2^7 to $N = 2^{12}$, compared to a 41-fold increase with full attention.

Furthermore, we compare the FLOPs across all methods on two datasets, namely BasicMotions and UWaveGestureLibrary. We can observe that DARKER reduces FLOPs by over 30× relative to full attention, and 18× to RFA-pos and RFA-trig at $N = 2^{12}$ in Figure 10, which shows its potential for handling large-scale data efficiently.

5.3.2 Experiment on accuracy. To evaluate the effectiveness of DARKER, we report the accuracy of DARKER alongside all methods across 14 representative datasets, detailed in Table 2. The results show that DARKER achieves the highest average accuracy among all methods, trailing closely behind full attention. Meanwhile, the average rank of DARKER is only slightly worse than Rocket and full attention, while p-values indicating no significant differences among the three methods. However, it is observed that the accuracy of Rocket on large dataset SpokenArabicDigits is about 0.3 lower than most of transformer methods. In a nutshell, as evidenced by the supplementary experiment in Appendix A.6, both the accuracy and efficiency of Rocket are limited in large datasets, which aligns with existing findings on non-transformer approaches [12, 28]. Additionally, we observed *overflow* in 6 out of 14 datasets for RFA-pos and 8 out of 14 for RFA-trig, highlighting such datasets are prone to inaccurate time series analysis. In contrast, DARKER consistently exhibits both high accuracy and efficiency.

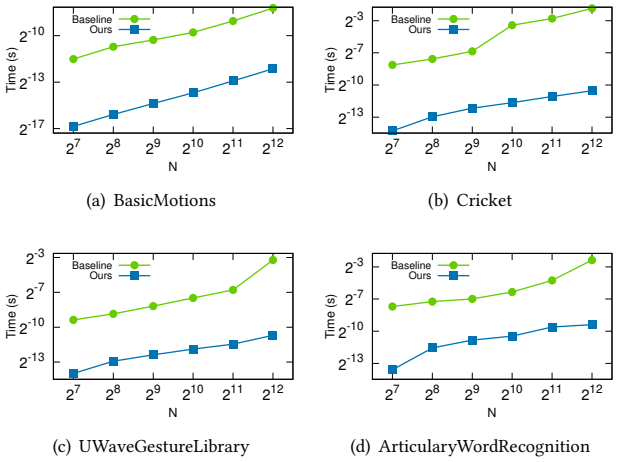
5.4 Experiment on pIndex

5.4.1 The efficiency of pIndex. To evaluate the efficiency of our pIndex for retrieving projections to S_θ, we conduct 100 queries on pIndex and calculate the average time. This is benchmarked against a baseline method of performing an ANN search on S using Euclidean Distance. The evaluation spanned four datasets, with N ranging from 2^7 to 2^{12}. According to Figure 11, our method consistently outperforms the baseline, being at least 20× faster, underscoring the effectiveness of pIndex. We also observe that the datasets with larger $\mathbb{N}$ (*e.g.*, ArticularyWordRecognition), the efficiency advantage of pIndex becomes even more significant, highlighting its capability in handling large-scale datasets.

5.4.2 Accuracy on the ratios for learning projections. To evaluate the impact of the size of $\mathcal{S}_\Phi$ handling in learning projection to the accuracy, we sample different ratios of $\mathcal{S}$ as $\mathcal{S}_\Phi$, and the rest of $\mathcal{S}$ is taken as $\overline{\mathcal{S}}_\Phi$. The experiment is conducted on four datasets, namely, ArticularyWordRecognition, BasicMotions, Cricket, and SelfRegulationSCP1. Figure 12 reveals a general trend: an increase in the ratio typically enhances our method's accuracy, aligning

Table 2: Accuracy of our method and 8 benchmarked methods on UEA datasets (An underline indicates an overflow, resulting in accuracy similar to random guesses)

	EDI	Rocket	RFA-pos	RFA-trig	Informer	Fedformer	Pyraformer	Full Attn	Ours
ArticularyWordRecognition	0.970	0.996	0.967	0.877	0.980	0.970	0.970	0.980	0.963
BasicMotions	0.676	0.990	0.250	0.250	1.000	1.000	0.975	1.000	1.000
CharacterTrajectories	0.964	N/A	0.060	0.060	0.962	0.974	0.970	0.972	0.976
Cricket	0.944	1.000	0.083	0.083	0.889	0.819	0.917	0.944	1.000
EigenWorms	0.549	0.863	0.420	0.420	0.710	0.756	0.420	0.732	0.565
Epilepsy	0.666	0.991	0.928	0.246	0.964	0.957	0.935	0.957	0.964
Handwriting	0.200	0.567	0.340	0.042	0.319	0.289	0.319	0.334	0.305
Libras	0.833	0.906	0.811	0.783	0.856	0.133	0.840	0.850	0.722
Phoneme	0.104	0.284	0.026	0.026	0.175	0.172	0.154	0.178	0.169
RacketSports	0.868	0.928	0.263	0.263	0.783	0.796	0.736	0.770	0.809
SelfRegulationSCP1	0.771	0.866	0.502	0.502	0.659	0.683	0.590	0.669	0.785
SelfRegulationSCP2	0.483	0.514	0.500	0.500	0.478	0.467	0.505	0.511	0.561
SpokenArabicDigits	0.967	0.630	0.100	0.100	0.972	0.968	0.971	0.973	0.979
UWaveGestureLibrary	0.881	0.944	0.888	0.125	0.881	0.903	0.897	0.900	0.906
Avg.Acc	0.705	0.749	0.438	0.306	0.760	0.706	0.727	0.769	0.765
Avg.Rank	5.714	2.393	7.143	8.214	4.464	4.857	5.393	3.357	3.464
Wilcoxon Test p-value	0.049	0.286	0.005	0.004	0.593	0.055	0.035	0.801	-



(a) BasicMotions (b) Cricket

(c) UWaveGestureLibrary (d) ArticularyWordRecognition

Figure 11: Query time of PINDEX on four datasets

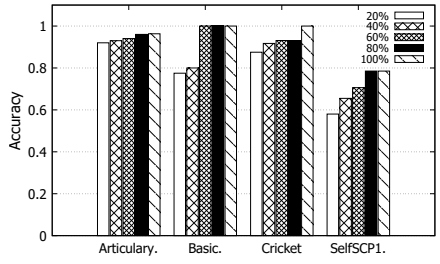
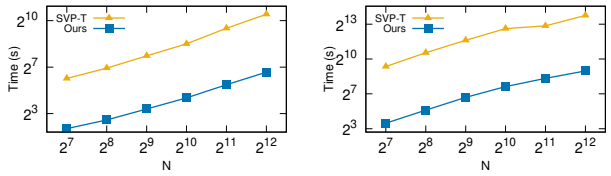


Figure 12: Accuracy on ratios for learning projections.

with Theorem 4.2 in Section 4.2. Meanwhile, for the BasicMotions dataset, the accuracy first increases when the ratio is from 20% to 60%, then stabilizes when the ratio is larger than 60%. This indicates the optimal accuracy can be attained without utilizing the full $\mathcal{S}$.



(a) BasicMotions (b) ArticularyWordRecognition

Figure 13: Efficiency on shape discovery of SVP-T, and ours

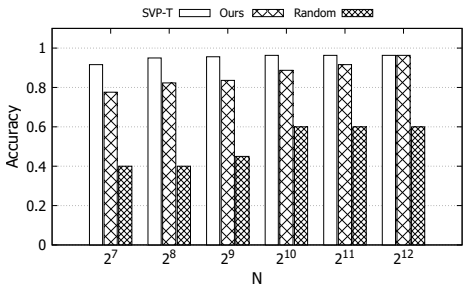


Figure 14: Accuracy on three shape discovery methods

5.5 Experiment on Shape Discovery

In this subsection, we evaluate our shape discovery method, as detailed in Section 4.3, against the preprocessing component of SVP-T, the latest method using shapes for a transformer model.

5.5.1 Efficiency of shape discovery. Figure 13 illustrates that our method's shape discovery process is approximately 10× faster than that of SVP-T across various N values on both datasets. Meanwhile, our DARKER exhibits a more pronounced superiority over SVP-T on ArticularyWordRecognition. This is due to the longer subsequences present in ArticularyWordRecognition, where our method efficiently retrieves distances between these subsequences and clustering centers, as outlined in Lines 9-10 of Algorithm 3. Our

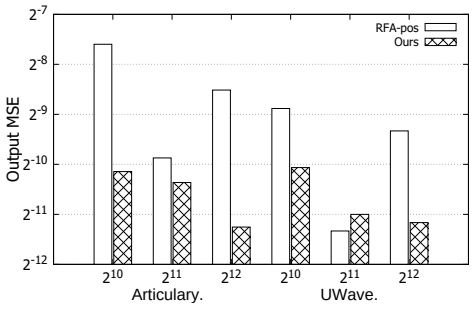


Figure 15: Approximation error to softmax **attention**

method demonstrates superior efficiency, particularly for datasets with longer time series subsequences, presenting its potential for large-scale time series analysis.

5.5.2 Overall Accuracy of Shape Discovery. We further conducted an experiment on the performance of our shape discovery method (Section 4.3) and the number of clustering centers (the number of shapes N) affect the accuracy of the dataset ArticularyWordRecognition. In addition to the most relevant work SVP-T, we present the accuracy of randomly selecting time series subsequences as shapes. As depicted in Figure 14, a larger N will generally lead to better accuracy across all three methods, and thus, we set $N = 2^{12}$. As a reference, the accuracy of a transformer with randomly selected time series subsequences as input does not exceed 0.6, which is significantly lower than the accuracy of the other two methods. In contrast, the accuracy of our method is nearly the same as SVP-T, while ours is on average 10× faster as shown in Figure 13.

5.6 Approximation Error

We investigate the approximation error of DARKER in contrast to RFA method across different attention sizes, by varying N on the ArticularyWordRecognition and UWaveGestureLibrary datasets. We choose RFA-pos as the benchmark due to its generally lower approximation error compared to RFA-trig [5]. As depicted in Figure 15, the approximation errors of our method consistently remain below 2^{-10}, outperforming RFA-pos in all tested scenarios, with the only exception at $N = 2^{11}$ for the UWaveGestureLibrary dataset. The lower and more stable approximation errors highlights the superior performance of our method in approximating softmax attention, particularly for time series data.

6 RELATED WORK

In this section, we review some related works on efficient transformer and transformer-based methods for time series.

6.1 Efficient Transformers

Transformer model has found broad applications not only in natural language processing [9, 47, 56] and computer vision [10, 26], but also in time series [45, 58, 61, 63]. Consequently, a research trend has emerged, focusing on improving transformer efficiency [27]. Interested readers can refer to a recent survey paper [43].

In this subsection, we introduce some relevant works of efficient transformers. Reformer [17] employs locality-sensitive hashing (LSH) attention that focuses on a small subset of keys for each query. However, it assumes that the Q and K matrices are identical, a prior, which is too restrictive in practical scenarios. Cluster-former [48], in contrast, applies the K-means algorithm to cluster hidden states rather than using LSH. Informer [61], an efficient transformer variant designed for time series forecasting, exploits the prior of sparsity in the attention weights matrix to reduce the time complexity to $O(N \log N)$. Despite the improvements, the training of Informer is still time-consuming and nearly the same as vanilla transformer. More recently, the matrix approximation method has been proposed for improving the efficiency [43]. Linformer, for example, uses a low-rank matrix to approximate attention, achieving both memory and time efficiency [49]. However, we find that these efficient transformers require priors, which may not hold in time series data.

Kernel-based methods [5, 35, 37, 60], are also referred to as random feature attention (RFA) methods or kernel-based methods in recent literature [35, 43]. We have adopted the term RFA in this paper. These methods redefine the attention mechanism by representing the exponential operation in attention as a kernel function, and need no priors. They aim to find a projection, ϕ, with random feature to approximate the exponential kernel without any priors. However, ϕ is defined as a fixed projection (*e.g.*, Equation 8), which cannot capture the input distribution. As a result, this leads to three challenges when applying kernel-based methods to time series, as have detailed in Section 3.

6.2 Transformer-based Methods for Time Series

Recently, transformer model has been introduced to many time series tasks, including forecasting [25, 30, 61], imputation [31, 59], anomaly detection [46, 54], and classification [55, 58, 63], which have demonstrated the superior performance [52]. Since we present our techniques with TSC, here, we introduce some TSC methods using a transformer model. Zerveas et al. [58] utilized the transformer for time series representation learning with classification as one of the downstream tasks. Their approach, treating the value at each timestamp as an input for the transformer, requires high training time when handling long time series. In contrast, TARNet [6] leverages the transformer to mask the timestamp and conduct reconstruction for TSC. SVP-T [63] takes the shapes discovered through clustering as input to the transformer, enabling the processing of time series of any length. However, transformer-based methods for TSC still necessitate quadratic computational complexity.

7 CONCLUSION

In conclusion, this is the first work undertaking the approach that utilizes a series of machine learning models, and an index to improve the efficiency of a time-consuming transformer model. Specifically, in DARKER, we propose a data-driven kernel-based attention that reduces the time complexity from quadratic to linear. In addition, we construct a PINDEX based on cluster information to efficiently search projection. Finally, we build an inverted index to optimize the shape discovery. The experiment shows that our method is more efficient than all compared methods and achieve nearly same average rank in terms of accuracy compared to vanilla attention. In the future, we plan to apply DARKER to optimize other downstream tasks of time series, such as time series clustering, and reducing the I/O time in GPU shown in Appendix A.4.

REFERENCES

[1] Anthony Bagnall, Hoang Anh Dau, Jason Lines, Michael Flynn, James Large, Aaron Bostrom, Paul Southam, and Eamonn Keogh. 2018. The UEA multivariate time series classification archive, 2018. *arXiv preprint arXiv:1811.00075* (2018).

[2] Saloman Bochner. 1955. *Harmonic Analysis and the Theory of Probability*. University of California Press, Berkeley.

[3] Angela Bonifati, Francesco Del Buono, Francesco Guerra, Miki Lombardi, and Donato Tiano. 2023. Interpretable Clustering of Multivariate Time Series with Time2Feat. *PVLDB* (2023), 3994–3997.

[4] 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.

[5] 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*.

[6] Ranak Roy Chowdhury, Xiyuan Zhang, Jingbo Shang, Rajesh K. Gupta, and Dezhi Hong. 2022. TARNet: Task-Aware Reconstruction for Time-Series Transformer. In *ACM SIGKDD*. 212–220.

[7] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. In *Advances in Neural Information Processing Systems*.

[8] Angus Dempster, François Petitjean, and Geoffrey I Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (2020), 1454–1495.

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

[10] 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*.

[11] 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.

[12] Navid Mohammadi Foumani, Chang Wei Tan, Geoffrey I Webb, and Mahsa Salehi. 2024. Improving position encoding of transformers for multivariate time series classification. *Data Mining and Knowledge Discovery* 38, 1 (2024), 22–48.

[13] Josif Grabocka, Nicolas Schilling, Martin Wistuba, and Lars Schmidt-Thieme. 2014. Learning time-series shapelets. In *ACM SIGKDD*. 392–401.

[14] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *ICML*. 5156–5165.

[16] Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR*.

[17] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. 2021. Reformer: The efficient transformer. In *ICLR*.

[18] S Knieling, J Niediek, E Kutter, J Bostroem, CE Elger, and F Mormann. 2017. An online adaptive screening procedure for selective neuronal responses. *Journal of neuroscience methods* 291 (2017), 36–42.

[19] 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.

[20] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. 2019. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE TKDE* 32, 8 (2019), 1475–1488.

[21] Jason Lines, Luke M Davis, Jon Hills, and Anthony Bagnall. 2012. A shapelet transform for time series classification. In *ACM SIGKDD*. 289–297.

[22] Hanxiao Liu, Zihang Dai, David So, and Quoc V Le. 2021. Pay attention to mlps. *NeurIPS* 34 (2021), 9204–9215.

[23] Meng Liu, Hongyang Gao, and Shuiwang Ji. 2020. Towards deeper graph neural networks. In *ACM SIGKDD*. 338–348.

[24] Shizhan Liu, Hang Yu, Cong Liao, Jianguo Li, Weiyao Lin, Alex X Liu, and Schahram Dustdar. 2021. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International conference on learning representations*.

[25] Yong Liu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Non-stationary Transformers: Exploring the Stationarity in Time Series Forecasting. In *NeurIPS*.

[26] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*.

[27] Xupeng Miao, Yujie Wang, Youhe Jiang, Chunan Shi, Xiaonan Nie, Hailin Zhang, and Bin Cui. 2022. Galvatron: Efficient transformer training over multiple gpus using automatic parallelism. *PVLDB* (2022), 470–479.

[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] 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*.

[31] Eunkyu Oh, Taehun Kim, Yunhu Ji, and Sushil Khyalia. 2021. STING: Self-attention based Time-series Imputation Networks using GAN. In *IEEE ICDM*, James Bailey, Pauli Miettinen, Yun Sing Koh, Dacheng Tao, and Xindong Wu (Eds.). 1264–1269.

[32] Stephen M Omohundro. 1989. *Five balltree construction algorithms*. International Computer Science Institute Berkeley.

[33] Themis Palpanas. 2015. Data series management: The road to big sequence analytics. *ACM SIGMOD Record* 44, 2 (2015), 47–52.

[34] John Paparrizos and Sai Prasanna Teja Reddy. 2023. Odyssey: An Engine Enabling The Time-Series Clustering Journey. *PVLDB* (2023), 4066–4069.

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

[36] Yun Peng, Byron Choi, Tsz Nam Chan, Jianye Yang, and Jianliang Xu. 2023. Efficient Approximate Nearest Neighbor Search in Multi-dimensional Databases. *SIGMOD* 1, 1 (2023), 1–27.

[37] Zhen Qin, Xiaodong Han, Weixuan Sun, Dongxu Li, Lingpeng Kong, Nick Barnes, and Yiran Zhong. 2022. The Devil in Linear Transformer. In *ACM EMNLP*. 7025–7041.

[38] Ali Rahimi and Benjamin Recht. 2007. Random features for large-scale kernel machines. *NeurIPS* (2007).

[39] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. 2021. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 35, 2 (2021), 401–449.

[40] Syed Yousaf Shah, Dhaval Patel, Long Vu, Xuan-Hong Dang, Bei Chen, Peter Kirchner, Horst Samulowitz, David Wood, Gregory Bramble, Wesley M. Gifford, Giridhar Ganapavarapu, Roman Vaculín, and Petros Zerfos. 2021. AutoAI-TS: AutoAI for Time Series Forecasting. In *ACM SIGMOD*. 2584–2596.

[41] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 56 (2014), 1929–1958.

[42] Emmanouil Sylligardos, Paul Boniol, John Paparrizos, Panos E. Trahanias, and Themis Palpanas. 2023. Choose Wisely: An Extensive Evaluation of Model Selection for Anomaly Detection in Time Series. *PVLDB* (2023), 3418–3432.

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

[44] Donato Tiano, Angela Bonifati, and Raymond Ng. 2021. FeatTS: Feature-based Time Series Clustering. In *ACM SIGMOD*, Guoliang Li, Zhanhuai Li, Stratos Idreos, and Divesh Srivastava (Eds.). 2784–2788.

[45] 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.

[46] 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.

[47] 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*.

[48] Ningning Wang, Guobing Gan, Peng Zhang, Shuai Zhang, Junqiu Wei, Qun Liu, and Xin Jiang. 2022. ClusterFormer: Neural Clustering Attention for Efficient and Effective Transformer. In *ACM ACL*.

[49] Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768* (2020).

[50] Qingsong Wen, Kai He, Liang Sun, Yingying Zhang, Min Ke, and Huan Xu. 2021. RobustPeriod: Robust time-frequency mining for multiple periodicity detection. In *ACM SIGMOD*. 2328–2337.

[51] Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. 2023. Transformers in Time Series: A Survey. In *IJCAI*. 6778–6786.

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

[53] Xinle Wu, Dalin Zhang, Miao Zhang, Chenjuan Guo, Bin Yang, and Christian S. Jensen. 2023. AutoCTS+: Joint Neural Architecture and Hyperparameter Search for Correlated Time Series Forecasting. *PVLDB* (2023), 97:1–97:26.

[54] Jiehui Xu, Haixu Wu, Jianmin Wang, and Mingsheng Long. 2022. Anomaly transformer: Time series anomaly detection with association discrepancy. *ICLR* (2022).

[55] Chao-Han Huck Yang, Yun-Yun Tsai, and Pin-Yu Chen. 2021. Voice2series: Reprogramming acoustic models for time series classification. In *ICML*. 11808–11819.
[56] Dezhong Yao, Yuhong Gu, Gao Cong, Hai Jin, and Xinqiao Lv. 2022. Entity Resolution with Hierarchical Graph Attention Networks. In *ACM SIGMOD*, Zachary G. Ives, Angela Bonifati, and Amr El Abbadi (Eds.). 429–442.
[57] Yuanyuan Yao, Dimeng Li, Hailiang Jie, Lu Chen, Tianyi Li, Jie Chen, Jiaqi Wang, Feifei Li, and Yunjun Gao. 2023. SimpleTS: An Efficient and Universal Model Selection Framework for Time Series Forecasting. *PVLDB* (2023), 3741–3753.
[58] 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.
[59] Jingqi Zhao, Chuitian Rong, Chunbin Lin, and Xin Dang. 2023. Multivariate time series data imputation using attention-based mechanism. *Neurocomputing* 542 (2023), 126238.
[60] Lin Zheng, Chong Wang, and Lingpeng Kong. 2022. Linear complexity randomized self-attention mechanism. In *ICML*. 27011–27041.
[61] 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.
[62] 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.
[63] 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.

Table 3: Details of datasets

Dataset name	No.Instance	Length
BasicMotions	40	100
Cricket	108	1197
UWaveGestureLibrary	120	315
EigenWorms	128	17984
Epilepsy	137	206
Handwriting	150	152
RacketSports	151	30
Libras	180	45
SelfRegulationSCP2	200	1152
SelfRegulationSCP1	268	896
ArticularyWordRecognition	275	144
CharacterTrajectories	1422	1422
Phoneme	6599	3315
SpokenArabicDigits	6599	6599
Synthetic A	10000	1024
Synthetic B	20000	1024

Table 4: Hyperparameters

Dataset name	Batch size	Epochs
BasicMotions	32	100
Cricket	4	200
UWaveGestureLibrary	32	100
EigenWorms	4	100
Epilepsy	4	200
Handwriting	4	200
RacketSports	32	200
Libras	4	100
SelfRegulationSCP2	4	200
SelfRegulationSCP1	64	100
ArticularyWordRecognition	64	200
CharacterTrajectories	8	200
Phoneme	64	100
SpokenArabicDigits	4	200
Synthetic A	8	200
Synthetic B	8	200

A APPENDIX

A.1 Details of datasets

We provide the size of training set (instance number) and time series length for the 14 real datasets and 2 synthetic datasets shown in Table 3. Readers may refer to the UEA archive paper [1] for more information about 14 datasets. For Synthetic A and B, we use numpy to randomly generate them.

A.2 All results of training time

We provide the experimental results on the efficiency of DARKER for the rest 8 datasets not presented in Section 5.3. As shown in Figure 16 and Figure 17, we observed that the trends for these datasets are consistent with the experimental results in Section 5.3, and DARKER continues to be 3×-4× faster than vanilla transformer and 1.5×-3× faster than other SOTAs for long sequences.

A.3 Details of hyperparameters

For training the transformer model, we follow the previous work [58, 63] to split the training set into two parts, namely 80% and 20%, where the 20% part is as the validation set to tune the hyperparameters. Then, the model is trained on the whole training set based on the tuned hyperparameters. The details of hyperparameters of all datasets are shown in Table 4.

A.4 Reducing the I/O TIME

For future work, the efficiency of DARKER can be further improved by reducing the I/O time when loading data in GPU. This could be done by using the matrix tilling in Flash attention [7].

A.5 Detail proofs

A.5.1 Proof of Lemma 4.1.

Proof. We take Frobenius norm into our computation corresponding to $dist(\cdot,\cdot)$ which implies $\|S_\theta - S_i\| = \|(S_\theta - S_i)^\top\|$. Assume that S_i and S_θ approximately follow a uniform distribution in Euclidean space after random sampling. Make $\sigma_n = dist(S_n, S_\theta) = \|S_\theta - S_n\|$, given $S_i \in \mathcal{S}_\Phi$ and for any $S_j \in \overline{\mathcal{S}}_\Phi$, we have:

$$\Pr[\sigma_j \geq \sigma_i] = 1 - O[(\frac{\sigma_i}{\max(\sigma_i)})^{Nd}] \tag{21}$$

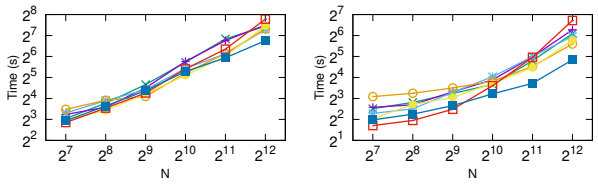
Assume that $S_j \to S_\theta$, *i.e.*, $\sigma_j \to 0$, we have:

$$\Pr[\sigma_i \leq \varepsilon_0] = 1 - O(p^{Nd}) \tag{22}$$

where $\varepsilon_0 = o(1)$ and $p = \frac{\sigma_i}{\max(\sigma_i)} \in (0, 1]$. With Equation 13, we have:

$$\begin{aligned} &\|S_\theta W_S S_\theta^\top - S_i W_S S_i{}^\top\| \\ &= \|S_\theta W_S S_\theta^\top - S_i W_S S_\theta^\top + S_i W_S S_\theta^\top - S_i W_S S_i{}^\top\| \\ &= \|(S_\theta - S_i) W_S S_\theta^\top + S_i W_S (S_\theta - S_i)^\top\| \\ &\leq \|S_\theta - S_i\| \|W_S S_\theta^\top\| + \|S_i W_S\| \|(S_\theta - S_i)^\top\| \\ &= \sigma_i \|W_S S_\theta^\top\| + \sigma_i \|S_i W_S\| \end{aligned} \tag{23}$$

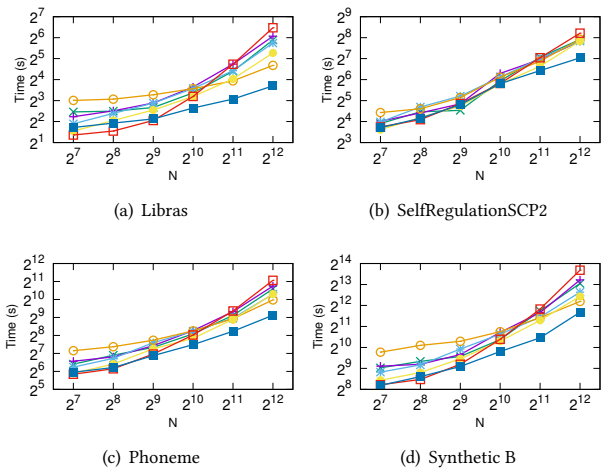
Since the last step in Equation 23 still contains S_θ, we transform the step into a combination of known variables as below:



(a) EigenWorms (b) Epilepsy

(c) Handwriting (d) RacketSports

Figure 16: Other results of training time (Part A)



(a) Libras (b) SelfRegulationSCP2

(c) Phoneme (d) Synthetic B

Figure 17: Other results of training time (Part B)

$$
\begin{aligned}
&\|S_\theta W_S S_\theta^\top - S_i W_S S_i{}^\top\| \\
&< \sigma_i \|W_S S_\theta^\top - W_S S_i^\top + W_S S_i^\top\| + \sigma_i \|S_i W_S\| \\
&\leq \sigma_i \|W_S (S_\theta - S_i)^\top\| + \sigma_i \|W_S S_i^\top\| + \sigma_i \|S_i W_S\| \\
&= \sigma_i^2 \|W_S\| + \sigma_i \|W_S S_i^\top\| + \sigma_i \|S_i W_S\| = \varepsilon
\end{aligned} \tag{24}
$$

With Equation 22, we achieve:

$$
\Pr[\|S_\theta W_S S_\theta^\top - S_i W_S S_i{}^\top\| < \varepsilon] \geq 1 - O(p^{Nd}) \tag{25}
$$

□

The probability in Equation 25 is no less than that in Equation 22. Specifically, we make a linear transformation based on $\|S_\theta - S_i\|$ and obtain inequation (Equation 23) that maintains the inequality continuity.

A.5.2 Proof of Theorem 4.2.

Proof. We continue with the notation from Lemma 4.1 here for convenience. Let $x_i = S_i W_S S_i^\top$, $x_\theta = S_\theta W_S S_\theta^\top$. With Lemma 4.1, we can achieve $\Pr[\|x_\theta - x_i\| < \varepsilon] \geq 1 - O(p^{Nd})$. Universal approximation theorem shows a unified form of the trained function [14], which implies that Φ_i and Φ_θ are continuous functions satisfying the local Lipsitz continuity condition. Formally, we have:

$$
\|\Phi_\theta(x_\theta) - \Phi_\theta(x_i)\| \leq L_\theta \varepsilon,\ \|\Phi_i(x_\theta) - \Phi_i(x_i)\| \leq L_i \varepsilon \tag{26}
$$

where L_θ and L_i are two positive Lipsitz constants. As a result, we obtain:

$$
\begin{aligned}
\Pr[\|\Phi_\theta(x_\theta) - \Phi_\theta(x_i)\| \leq L_\theta \varepsilon] &\geq 1 - O(\varepsilon_0^{Nd}) \\
\Pr[\|\Phi_i(x_\theta) - \Phi_i(x_i)\| \leq L_i \varepsilon] &\geq 1 - O(\varepsilon_0^{Nd})
\end{aligned} \tag{27}
$$

where L_θ and L_i are two positive constants.

Make $SM(x) = \text{softmax}(x)$, $\eta_n = \|\Phi_n(x_i) - SM(x_i)\|$, for a data-driven $\Phi_{j \neq i, j \in \mathbb{Z} \cap [1,N]}$ we have:

$$
\Pr[\|\Phi_j(x_i) - SM(x_i)\| > \eta_i] = 1 - O[(\frac{\eta_i}{\max(\eta_j)})^{Nd}] \tag{28}
$$

For Φ_θ, we have:

$$
\Pr[\|\Phi_\theta(x_i) - SM(x_i)\| \leq \eta_i] = [1 - O(q^{Nd})]^{N-1} \tag{29}
$$

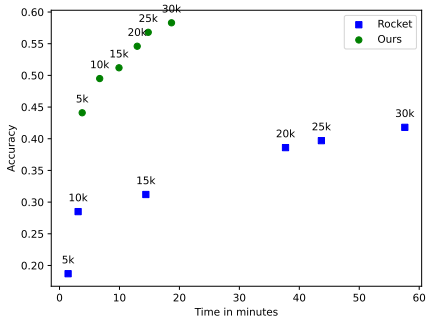


Figure 18: Comparison the runtime and accuracy of DARKER and Rocket on the largest dataset in UEA InsectWingBeat.

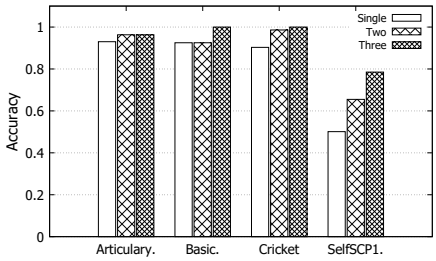


Figure 19: Accuracy of different numbers of window sizes.

where $q = \frac{\eta_i}{\max(\eta_j)} \in (0, 1)$. According to the triangle inequation, we know:

$$
\begin{aligned}
&\|\Phi_\theta(x_\theta) - \Phi_i(x_\theta)\| \\
&= \|\Phi_\theta(x_\theta) - \Phi_\theta(x_i) + (\Phi_\theta(x_i) - SM(x_i)) - \\
&\quad (\Phi_i(x_i) - SM(x_i)) + (\Phi_i(x_i) - \Phi_i(x_\theta))\| \\
&\leq \|\Phi_\theta(x_\theta) - \Phi_\theta(x_i)\| + \|\Phi_\theta(x_i) - SM(x_i)\| + \\
&\quad \|\Phi_i(x_i) - SM(x_i)\| + \|\Phi_i(x_i) - \Phi_i(x_\theta)\|
\end{aligned} \tag{30}
$$

Finally, we achieve:

$$
\begin{aligned}
&\Pr[\|\Phi_\theta(x_\theta) - \Phi_i(x_\theta)\| < (L_\theta + L_i)\varepsilon + 2\eta_i] \\
&\geq \Pr[\|\Phi_\theta(x_\theta) - \Phi_\theta(x_i)\| \leq L_\theta \varepsilon] \cdot \Pr[\|\Phi_i(x_\theta) - \Phi_i(x_i)\| \leq L_i \varepsilon] \\
&\quad \cdot \Pr[\|\Phi_\theta(x_i) - SM(x_i)\| \leq \eta_i] \\
&\geq [1 - O(p^{Nd})]^2 \cdot [1 - O(q^{Nd})]^{N-1}
\end{aligned} \tag{31}
$$

□

A.6 Compared with Rocket

To further compare our method with Rocket on a large dataset, we follow the experiment in [12], and use the same dataset, InsectWingBeat, the largest UEA datasets [1]. The result is consistent with [12] as shown in Figure 18. We observe that when the size of training set is only $5k$ or $10k$, Rocket is faster than DARKER. However, DARKER becomes faster than Rocket when the size is larger than $15k$. Moreover, DARKER (under the default setting) is almost three times faster when the size is $30k$. This result confirms that DARKER is more suitable for solving large time series datasets.

A.7 Sliding Windows Size

We have conduct experiments to evaluate the effectiveness of using single size sliding windows, dual-size combinations, and triple-size combinations on four separate datasets, ArticularyWordRecognition, BasicMotions, Cricket, and SelfRegulationSCP1. The results in Figure 19 show that applying three different sizes of sliding windows usually yields better performance, which is consistent with findings in other related works [13, 19, 63]. Thus, we follow existing works [19, 63] to use three sliding window sizes. The sizes are $\{0.1t, 0.2t, 0.3t\}$, where t is the length of the time series.