# SEAnet: A Deep Learning Architecture for Data Series Similarity Search

Qitong Wang and Themis Palpanas

***Abstract*—A key operation for massive data series collection analysis is similarity search. According to recent studies, SAX-based indexes offer state-of-the-art performance for similarity search tasks. However, their performance lags under high-frequency, weakly correlated, excessively noisy, or other dataset-specific properties. In this work, we propose Deep Embedding Approximation (DEA), a novel family of data series summarization techniques based on deep neural networks. Moreover, we describe SEAnet, a novel architecture especially designed for learning DEA, that introduces the Sum of Squares preservation property into the deep network design. We further enhance SEAnet with SEAtrans encoder. Finally, we propose novel sampling strategies, SEAsam and SEAsamE, that allow SEAnet to effectively train on massive datasets. Comprehensive experiments on 7 diverse synthetic and real datasets verify the advantages of DEA learned using SEAnet in providing high-quality data series summarizations and similarity search results.**

***Index Terms*—Data series, similarity search, neural networks, sampling.**

## I. Introduction

With the rapid developments and deployments of modern sensors, massive data series[1] datasets are now being generated, collected and analyzed in almost every scientific domain [1]. Typical data series analysis techniques are querying [2], classification [3], clustering [4], anomaly detection [5], and visualization [6], for all of which similarity search plays a central role. Data series similarity search aims to find the closest series in a dataset to a given query series according to a distance measure, such as euclidean distance, which is one of the most widely used [7]. Similarity search can be divided into exact search and approximate search [8]. Approximate similarity search may not always produce the exact answers, but in most cases it produces answers that are very close to the exact ones [2]. Thus, it is widely used on massive series collections to enable interactive data exploration and other latency-bounded applications [9]. In this work, we focus on approximate similarity search under euclidean distance.

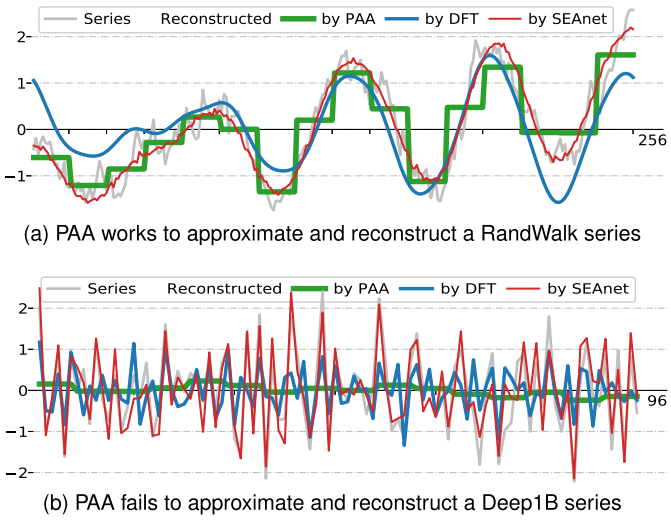


Fig. 1. Case studies where PAA and DFT work or fail to approximate and reconstruct series from RandWalk and Deep1B datasets. In both cases, DEA works to approximate and reconstruct series. All summarizations use the same memory budget.

Indexes are widely employed to speed up data series similarity search [2], [8]. Most indexes are based on summarized representations of the data series [7] of lower dimensionality.[2] Symbolic Aggregate approXimation (SAX) [10] is a popular and effective discretized summarization. SAX-based indexes [11] are the state-of-the-art (SOTA) data series similarity search methods [2], [8].

Nevertheless, SAX-based indexes suffer from the problem that SAX fails in hard datasets with specific properties [12]. Since SAX is the symbolization of Piecewise Aggregate Approximation (PAA) [10], failure of PAA to correctly represent some data series directly translates to failure of the PAA-based SAX. Fig. 1 illustrates a working and a failing case. The high frequency of the Deep1B series (Fig. 1(b)) implies more periodic intervals than the available SAX words: each PAA segment has to average values over $\geq 1$ intervals, leading to similar PAA values across different segments, and to indistinguishable SAX words across different series. Introducing more SAX words

Manuscript received 16 April 2022; revised 8 January 2023; accepted 8 April 2023. Date of publication 25 April 2023; date of current version 8 November 2023. This work was supported in part by IdEx Université Paris Cité under Grant ANR-18-IDEX-0001, and in part by China Scholarship Council, FMJH Program PGMO, HIPEAC 4, GENCI–IDRIS under Grants 2020-101471, 2021-101925, and 2022-AD011012641R, and NVIDIA Corporation for the Titan Xp GPU donation used in this research. Recommended for acceptance by Z. Wang. *(Corresponding author: Qitong Wang.)*

The authors are with LIPADE, Université Paris Cité, 75006 Paris, France (e-mail: qitong.wang@etu.u-paris.fr; themis@mi.parisdescartes.fr).

Digital Object Identifier 10.1109/TKDE.2023.3270264

[1]A data series, or data sequence, is an ordered sequence of points. The most common type of data series is time series, where the dimension that imposes the sequence ordering is time; though, this dimension could also be mass, angle, or position [1].

[2]In the data-series literature, *dimensionality* is used interchangeably with *length*, to refer to the number of values of a univariate series.

1041-4347 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

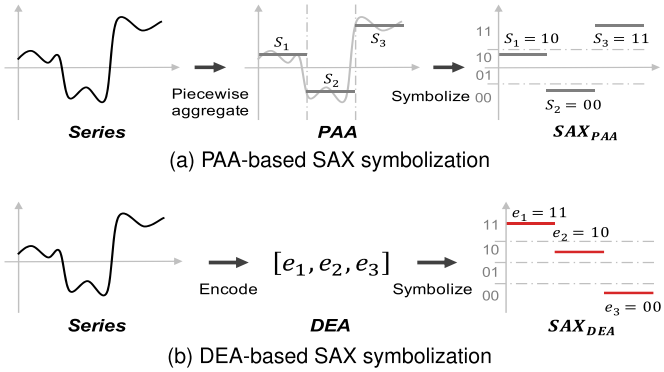


Fig. 2. Replace PAA by DEA for SAX symbolization.

could alleviate the problem, but would lead to an undesirably long summarization that could not be effectively indexed.

In this work, we propose to build a data series index based on *D*eep *E*mbedding *A*pproximations (DEA), i.e., data series summarizations derived from embeddings learned using deep neural networks. Embedding techniques, or representation learning, is to learn vectors possessing necessary latent information for downstream applications. Its success in data series, as well as high-dimensional vectors has been reported in many applications [13]. Embedding techniques have been proven to be capable of capturing frequency [14] and other latent properties. However, to the best of our knowledge, data series embedding has not been adapted to and evaluated for similarity search.

In the case of data series similarity search, DEA may replace PAA, and then be symbolized and indexed by an iSAX index as illustrated in Fig. 2. DEA targets to preserve original pairwise distances in the lower-dimensional DEA space. Thus, it is naturally capable of being symbolized into SAX, on which an iSAX index can be built. Our work shows that compared to PAA and (PAA-based) SAX, DEA better preserves pairwise distances, leading to a more effective index for data series similarity search.

To effectively learn DEA on massive series collections, we propose a novel autoencoder architecture SEAnet (*SE*ries *A*pproximation *net*work). SEAnet's basic structure follows a full-preactivation ResNet [15]. It adopts the idea of exponentially increasing dilations, which has been verified to be efficient for data series applications [16]. Moreover, we further design a new SEAtrans encoder, enhanced by Transformer blocks [17] to improve the fixed global dependence framed by deeper dilated layers. In contrast to existing convolutional autoencoders for series embedding [18], SEAnet comprises both an encoder and a decoder. We argue (and experimentally verify) that a decoder is necessary to learn high-quality DEA for similarity search. Moreover, SEAnet is the first architecture to formally introduce the principle of *S*um *o*f *S*quares (SoS) preservation, and incorporate it into the network design. SoS preservation aims to keep the sum of squared values invariant throughout the transformations. We observe that defining new axes based on the largest SoS is equivalent to selecting the largest eigenvalues in eigenvalue-based linear dimensionality reductions on z-normalized datasets (i.e., mean=0, stddev=1) [19]. They both aim at preserving the largest variances in the dataset through linear transformations. In this sense, SoS could be regarded as an indicator of the quality of the transformation to a reduced dimensionality space performed by SEAnet (or other deep network architectures). Hence, we introduce SoS as an invariant to regularize SEAnet and other networks, and demonstrate its benefits.

Finally, we observe that training a deep neural network on very large sequence collections is prohibitively expensive. Thus, for efficient training, we propose SEAsam (SEA-sampling), a novel sampling strategy based on a sortable data series summarization [20]. SEAsam enables SEAnet (and other networks) to effectively fit a large dataset, leading to improved performance. Moreover, based on the observation that the raw data series are not the only sampling space to be preserved, we extend SEAsam by SEAsamE. SEAsamE offers to SEAnet representative training samples covering all three major sampling spaces, i.e., the raw data series, the data series pairs, and the reconstruction errors.

Comprehensive experiments verify that, compared to PAA and DEA generated by other SOTA architectures (FDJNet [18], TimeNet [21] and InceptionTime [3]), the DEA generated by SEAnet is more effective in preserving the original pairwise distances in the lower-dimensional summarized space. This advantage also leads to more accurate approximate similarity search across several synthetic and real data series collections with diverse properties.

*Contributions:* Our contributions[3] are as follows.

1) We propose the use of deep learning embeddings for data series similarity search. We introduce novel Deep Embedding Approximations, and show how these can be used to index the original data series and then support (approximate) similarity search queries. Our results can be used as a blueprint to facilitate further progress in this area.
2) We propose SEAnet, a novel architecture that is specifically built to support high-quality DEA and similarity search. SEAnet incorporates modern architectural elements designed for data series applications, including a full-preactivation ResNet and exponentially increasing dilations. We extend SEAnet with a new SEAtrans encoder to provide learnable global dependence for deeper layers.
3) We introduce and formalize the principle of Sum of Squares (SoS) preservation. SoS preservation is a general principle for any architecture to learn high-quality DEA for dimensionality reduction. We explain how it can benefit the DEA architectures (including SEAnet), and how to incorporate it into the architecture designs.
4) We propose SEAsam, a novel sampling strategy for massive data series collections, enabling effective training for the deep models. SEAsamE further extends SEAsam by exploiting three major sampling spaces in DEA learning and facilitates deep model training.
5) We also describe alternative deep architectures for DEA, based on the SOTA designs of FDJNet, TimeNet, and InceptionTime. We explain how our ideas can be applied on these architectures, and study in detail their performance.

[3]Compared to the previous conference version [22], this paper describes an enhanced encoder architecture and a new sampling method.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

6) Comprehensive experiments on three synthetic datasets and four real-world datasets verified the effectiveness of DEA and SEAnet for data series summarization and approximate similarity search, outperforming traditional iSAX-based solutions, as well as three other SOTA RNN and CNN architectures for series embedding. Datasets, codes and pre-trained models are available online.[4]

## II. Related Work

*Data Series Indexes:* The most prominent data series indexing techniques can be categorized into graph-based indexes [23], inverted indexes [24], Locality Sensitive Hashing (LSH) [25], optimized scans [26], and tree-based indexes [10], [27]. Recent studies [2], [8] have demonstrated that the SAX-based indexes [11] achieve SOTA performance under several conditions.

SAX [10] is a discretized series summarization based on PAA. PAA first transforms the data series into $l$ real values, and then SAX quantizes each PAA value using discrete symbols. SAX quantization utilizes a scalar alphabet of size $a$. The number of bits used to encode $a$ is called the cardinality of SAX. iSAX [10] enables the comparison of SAXs of different cardinalities, which makes SAX indexable. iSAX2.0 and iSAX2+ [28] improve iSAX with a better node-splitting policy and a bulk-loading strategy, ADS+ [29] makes iSAX adaptive, ULISSE[30] supports to variable-length queries, DPiSAX [31] and Odyssey [32] make iSAX distributed, ParIS [33], MESSI [11] and SING [34] make iSAX highly concurrent, Dumpy [35] introduces an adaptive node splitting algorithm that leads to a multi-ary data structure, while Hercules [36] and Elpis [37] combine the iSAX and APACA [27] summarizations. Coconut-Trie and Coconut-Tree [38] transform SAX into a sortable summarization to enable more bulk loading opportunities.

In this work, we employ MESSI [11] as our indexing and query answering framework, because its memory-based concurrent design leads to SOTA performance. Moreover, since all SAX-based indexes share the same summarization techniques, improving the query answers quality of MESSI translates to the same improvements for all SAX-based indexes.

*Learned Data Series Embeddings:* we observe that few recent works [18], [21] focus on data series representation learning, none of which targets similarity search. Autoencoder is a category of deep neural networks to learn embeddings [18]. The encoder component of an autoencoder maps a dataset to lower dimensional vectors, i.e., embeddings; the decoder reverses this procedure. It has been empirically verified in many domains that embedding learns useful latent information [14].

TimeNet [21] and FDJNet [18] are two SOTA architectures for data series representation learning. TimeNet deploys a multi-layer GRU to embed and reconstruct series. FDJNet is based on Temporal Convolutional Network (TCN) to embed series. Aside from representation learning, deep models are exploited in other data series applications. The SOTA series classification method, InceptionTime [3] employs the Inception module for data series classification. However, neither TimeNet, FDJNet nor InceptionTime has been adapted and evaluated for similarity search before.

In contrast to all the above methods, the proposed SEAnet not only adopts design choices suitable for distance preservation, but also introduces a novel and general principle of SoS preservation for dimensionality reduction.

Other data series embedding techniques that do not use neural networks include RWS [39], SPIRAL [40], and GRAIL [41]; these are built upon various matrix factorization techniques, e.g., Kernel Principal Component Analysis (KPCA). However, their high time and space complexities (mostly $\mathcal{O}(mn^2)$), prevents such techniques from being deploying on massive collections with hundreds of millions of data series, which is our goal.

## III. Background

A *data series*, $S = \{p_1, \ldots, p_m\}$, is a sequence of points, where each point $p_i = (v_i, t_i)$, $1 \leq i \leq m$ is associated to a real value $v_i$ and a position $t_i$. The position corresponds to the order of this value in the sequence. We call $m$ the *length* of the series. $\mathcal{S}$ denotes a collection of data series, i.e., $\mathcal{S} = \{S_1, \ldots, S_n\}$. We call $n$ the *size* of the series collection.

A *summarization* $E = \{e_1, \ldots, e_l\}$ of a series $S$ is a lower, $l$-dimensional representation, which preserves some desired properties of $\mathcal{S}$. For similarity search, the target property is pairwise distance space structure of $\mathcal{S}$, i.e., $\forall S_i, S_j \in \mathcal{S}, d'(E_i, E_j) \approx d(S_i, S_j)$, where $E_i$, $E_j$ are summarizations of $S_i$, $S_j$, $d(\cdot,\cdot)$, and $d'(\cdot,\cdot)$ are distance measures in series and summarization spaces, respectively.

The *distance measure* $d$ deployed in our work is euclidean distance, which is a widely adopted and effective measure for data series similarity search [7]. $d'$ in the summarization space needs not be the same as $d$, e.g., for PAA, $d'(\cdot,\cdot) = \frac{\sqrt{m}}{\sqrt{l}} \times d(\cdot,\cdot)$. $d'$ for DEA is the same as PAA if it's scaled for SoS preservation. Otherwise, $d'(\cdot,\cdot) = d(\cdot,\cdot)$.

Given a query series $S_q$ of length $m$, a series collection $\mathcal{S}$ of size $n$ and length $m$, a distance measure $d$, *similarity search* targets to identify the series $S_c \in \mathcal{S}$ whose distance to $S_q$ is the smallest, i.e., $\forall S_o \in \mathcal{S}, S_o \neq S_c, d(S_c, S_q) \leq d(S_o, S_q)$. Instead of finding the exact answer $S_c$, *approximate similarity search* targets to find very fast a series $S'_c \in \mathcal{S}$, whose distance to $S_q$ is small, without guaranteeing it is the smallest. $\frac{d(S_c,S_q)}{d(S'_c,S_q)} \in (0, 1]$ is called $S'_c$'s *tightness*.

## IV. DEA-Based Similarity Search

In this section, we present the proposed DEA-based data series similarity search framework, including the SEAnet architecture. The complete workflow is illustrated in Fig. 3. Given a series collection, SEAsam first draws representative samples to train SEAnet. After SEAnet converges, it embeds all series into DEAs, which are further discretized into SAXs. Thus, DEA-based SAXs are used in an iSAX index, where approximate similarity search can be efficiently conducted.

SEAnet is a novel autoencoder proposed to learn high-quality DEA. Unlike FDJNet [18] and other SOTA convolutional architectures for series embedding, SEAnet is composed of both an

[4]https://helios.mi.parisdescartes.fr/\,themisp/seanet/

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

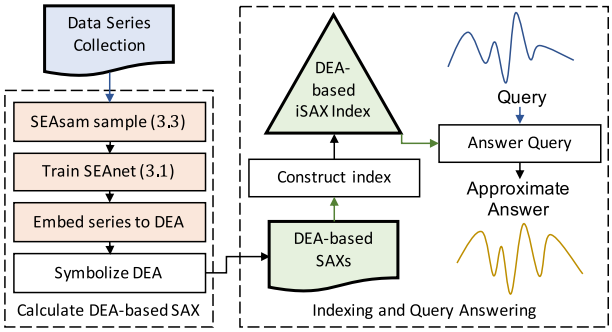


Fig. 3. Workflow of DEA-based approximate similarity search.

encoder and a decoder. The inclusion of the decoder is beneficial, since it can act as a regularizer to prevent SEAnet from falling into bad local optima, where DEAs become very similar to each other (and hence, not suitable for similarity search). SEAnet stacks dilated full-preactivation ResBlocks [15]. Its dilations increase exponentially with deeper layers. Moreover, SEAnet introduces the principle of SoS preservation for lower dimensionality representation learning. We present the details of SEAnet's design in Section IV-A, and further discuss the SoS preservation principle in Section IV-B.

To improve the fixed global dependence framed by the exponentially-increasing dilations, we propose a new SEAtrans encoder. SEAtrans encoder replaces SEAnet's deeper ResBlocks by Transformer blocks (TransBlocks) [17] to aggregate high-level information with learnable dependence. We present the SEAtrans encoder in Section IV-A.

Our SEAsam strategy makes use of the inverse iSAX sortable summarization [20]. In this scheme, SAX bits are interleaved, such that all significant bits across SAX words precede less significant bits, which renders the resulting representation, InvSax, sortable. This order has been shown to imply the distribution information of the dataset [20]. Thus, sampling proceeds by drawing series of equal intervals from dataset sorted in InvSAX order (cf. Section IV-C).

SEAsamE further extends SEAsam by exploiting three major sampling spaces for DEA learning, i.e., the spaces of raw data series, data series pairs and reconstruction errors. Hence, not only does it better represent the data series collection, but it also facilitates the convergence of SEAnet. We present SEAsamE in Section IV-C.

Since DEA acts as a replacement for PAA, indexing and query answering procedures remain similar to an ordinary iSAX. Considering the low-latency requirement of approximate similarity search applications, our design follows MESSI [11], the SOTA concurrent in-memory iSAX index.

### A. SEAnet Architecture

The SEAnet architecture is illustrated in Fig. 4(a). It comprises a convolutional encoder and a homogeneous decoder. We first overview the architecture and then present details.

The first part of the SEAnet encoder, from ConvLayer1 to MaxPool, comprises $k$ stacked dilated full-preactivation ResBlocks [15] for nonlinear transformations. The dilated ResBlock is illustrated separately in Fig. 4(b). Its dilation increases exponentially with the depth of the layer. Compared to constant dilations, this has been verified to effectively broaden the receptive fields for data series applications [16]. The dimension of latent vectors and number of channels are the same as the dimension of the input series. Thus, after MaxPooling within channels and squeezing, the first part could be regarded as an equi-length nonlinear transformation. The second part of the SEAnet encoder, from Linear1 to LayerNorm2, comprises two linear layers for dimensionality reduction. Unlike most existing encoders with linear final layers [18], the SEAnet encoder is finalized by LayerNorm2, which is specifically designed using the SoS preservation principle. We elaborate on this in Section IV-B.

The SEAnet decoder corresponds to the encoder, except for a preceding Tanh-activated linear layer, introduced to adjust dimensionality. We claim that the encoder and decoder need not be homogeneous. Although encoder-only architectures is the popular choice [18], we argue (and experimentally verify) that the decoder is necessary in similarity search applications in order to regularize the DEAs, so that they are distinguishable among each other. This results to a better indexing structure, and to a more effective and efficient similarity search.

*Extending SEAnet With SEAtrans Encoder:* As shown in Fig. 5, the SEAtrans encoder is composed by inserting $k_2$ stacked TransBlocks between the MaxPool and Linear1 layers. A TransBlock consists of one multiheaded self-attention layer and one feedforward layer [42]. The input tokens to the stacked TransBlocks are the ResBlocks' output vectors split by dimensions, concatenated by learnable positional encodings [43]. Each token corresponds to a dimension. Hence, each token represents the latent feature across all channels of each dimension. In addition, we append a special embedding token (EmbToken) as the learning target, following the convention of classification token [17].

The SEAtrans encoder complies to the SEAnet framework and is open to decoder architecture choices. SoS preservation (LayerNorm2 in Fig. 5) and SEAsam sampling also work for SEAtrans. We use the SEAtrans encoder with the the regular SEAnet decoder.

*Training Procedure:* We first provide the intuitions behind SEAnet training, and then the mathematical formalization.

SEAnet is trained pairwisely by mini-batched Stochastic Gradient Descent (SGD). Its loss function is a linear combination of two components: (1) The Compression Error $L_C$, i.e., the average differences between the original distance of data series pairs $(S_i, S_j)$ and their DEA distance. $L_C$ evaluates whether original distances are well-preserved in the DEA space. (2) The Reconstruction Error $L_R$, i.e., the average distance between the original series $S_i$ and the reconstructed series. $L_R$ evaluates how well the original series can be reconstructed using SEAnet. Moreover, we divide both the series and their DEAs by the square root of their lengths in $L_C$ and $L_R$. Together with the LayerNorm2 in SEAnet, this scaling not only preserves SoS for better dimensionality reduction, but also stabilizes the gradient propagation. The rationale behind these steps is further explained in Section IV-B.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

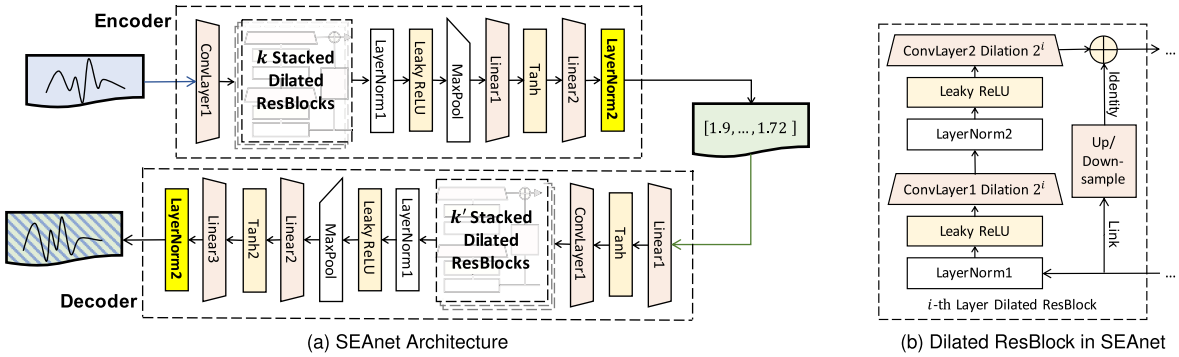


Fig. 4. The SEAnet architecture and the details of a dilated full-preactivation ResBlock.

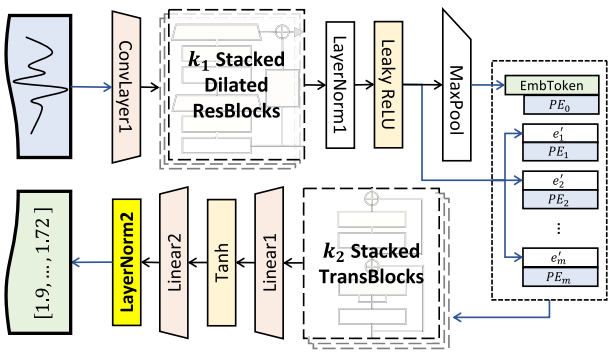


Fig. 5. The SEAtrans encoder architecture.

During each training epoch, for every series $S_i$ in the training set, a random but different series $S_j$ is drawn from the training set to form pairs $(S_i, S_j)$ for $L_C$. Since both $S_i$ and $S_j$ are from the training set, $S_j$ is detached (treated as constants instead of variables) to prevent its gradients from being back-propagated twice within one epoch.

We now formalize $L_C$ and $L_R$. First, we introduce the formula of SEAnet encoder as $E_i = \phi(S_i|\Theta_\phi)$ and decoder as $\widetilde{S}_i = \psi(E_i|\Theta_\psi) = \psi \cdot \phi(S_i|\Theta_{\phi,\psi})$, where $\phi$ and $\psi$ are mappings with parameters $\Theta_\phi$ and $\Theta_\psi$, $S_i$ is a series, $E_i$ is $S_i$'s DEA and $\widetilde{S}_i$ is $S_i$'s reconstruction. Without scaling, $L_C = \frac{1}{N_p}\sum_{(S_i,S_j)\in\mathcal{S}\times\mathcal{S}} |d(S_i, S_j) - d(\phi(S_i), \phi(S_j))|$, where $N_p$ is the number of sampled series pairs $(S_i, S_j)$, and $L_R = \frac{1}{N_s}\sum_{S_i\in\mathcal{S}} d(S_i, \psi \cdot \phi(S_i))$, where $N_s$ is the number of sampled series $S_i$. With scaling, $L_C = \frac{1}{N_p}\sum_{(S_i,S_j)\in\mathcal{S}\times\mathcal{S}} |\frac{1}{\sqrt{m}} d(S_i, S_j) - \frac{1}{\sqrt{l}} d(\phi(S_i), \phi(S_j))|$, and $L_R = \frac{1}{N_s}\sum_{S_i\in\mathcal{S}} \frac{1}{\sqrt{m}} d(S_i, \psi \cdot \phi(S_i))$. Thus, the loss function $L = L_C + \alpha L_R$, where $\alpha$ is a hyperparameter to balance between $L_C$ and $L_R$.

### B. Sum of Squares Preservation

We propose a SoS preservation framework for effective DEA learning. SoS preservation has been observed before [19], but to the best of our knowledge, has never been formally introduced to representation learning. Given an $n \times m$ matrix $M$, where row $M_{i,*}$ corresponds to a series and column $M_{*,j}$ corresponds to a position, SoS $= \sum_{i,j} M_{i,j}^2$. Note that in linear dimensionality reductions on z-normalized datasets, defining new axes based on the largest SoS is equivalent to selecting the largest eigenvalues, with the purpose of preserving information (measuerd by total variance) about the dataset through linear transformations [19]. Thus, SoS could be regarded as a useful indicator of the transformation quality. In SEAnet, by using SoS preservation as an invariant regularization, the quality of DEAs is upheld from this perspective, such that the neural network can focus on learning the nonlinear transformations.

Given the (z-normalized) input dataset, the proposed SoS preservation requires two steps: (1) z-normalizing the output of encoder (DEAs) and decoder (the reconstructed series); and (2) dividing the series and their DEAs by the square of their lengths in loss function $L$. Note that step (2) also takes the neural network convergence into consideration, as it benefits from the stabilization of the latent variables and variances [44]. We now elaborate on the design of SEAnet under this principle.

Considering the fact that z-normalizing data series is a very common operation [1], we constrain SEAnet to keep SoS invariant by forcing each DEA to preserve the SoS of its corresponding series. This is achieved by the following two steps: (1) z-normalizing the output of encoder, i.e., DEA; and (2) scaling DEA by $\sqrt{m/l}$. We formalize this idea below, and prove it by Lemma 1.

*Lemma 1:* Given a z-normalized series dataset $\mathcal{S}$ of size $n$ and its DEAs $\mathcal{E}$, $\mathcal{E}'$ is derived by z-normalizing and then multiplying $\mathcal{E}$ by $\frac{\sqrt{m}}{\sqrt{l}}$. $\mathcal{E}'$'s SoS is the same to $\mathcal{S}$, that is

$$\sum_{S_i\in\mathcal{S}} \sum_{p_j^i\in S_i} {p_j^i}^2 = \sum_{E_i\in\mathcal{E}} \sum_{e_j^i\in E_i} \left( \frac{\sqrt{m}}{\sqrt{l}} \frac{e_j^i - \overline{e^i}}{\sigma_{e^i}} \right)^2, \tag{1}$$

where $\overline{e^i}$ and $\sigma_{e^i}$ are the mean and standard deviation of DEA $e^i$ (without loss of generality, we assume $\sigma_{e^i} \neq 0$.).

*Proof:* First, as $\mathcal{S}$ is z-normalized with mean $= 0$ and variance $= 1$, the left side

$$\sum_{S_i\in\mathcal{S}} \sum_{p_j^i\in S_i} {p_j^i}^2 = \sum_{S_i\in\mathcal{S}} m \times \frac{\sum_{p_j^i\in S_i} (p_j^i - 0)^2}{m}$$
$$= \sum_{S_i\in\mathcal{S}} m \times 1 = nm.$$

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

For the right side of the equation,

$$\sum_{E_i\in\mathcal{E}}\sum_{e_j^i\in E_i}\left(\frac{\sqrt{m}}{\sqrt{l}}\frac{e_j^i-\overline{e^i}}{\sigma_{e^i}}\right)^2=\sum_{E_i\in\mathcal{E}}\sum_{e_j^i\in E_i}\frac{m}{l}\frac{(e_j^i-\overline{e^i})^2}{\sigma_{e^i}^2}$$
$$=\sum_{E_i\in\mathcal{E}}m\times\frac{1}{\sigma_{e^i}^2}\times\frac{\sum_{e_j^i\in E_i}(e_j^i-\overline{e^i})^2}{l}$$
$$=\sum_{E_i\in\mathcal{E}}m\times 1=nm.$$

Thus, the conclusion holds that

$$\sum_{S_i\in\mathcal{S}}\sum_{p_j^i\in S_i}{p_j^i}^2=nm=\sum_{E_i\in\mathcal{E}}\sum_{e_j^i\in E_i}\left(\frac{\sqrt{m}}{\sqrt{l}}\frac{e_j^i-\overline{e^i}}{\sigma_{e^i}}\right)^2.$$

*Scaling in Losses:* Scaling the DEAs raises another problem: its values will have a much larger variance. This hinders the convergence of the network, because of the internal covariate shift and other problems [44]. Latent variables with $\mu=0$ and $\sigma=1$ are widely considered among the best choices for the gradients' back-propagation [44]. Hence, we keep the DEAs z-normalized, and scale the series by $\sqrt{1/m}$ and DEA by $\sqrt{1/l}$ in $L_C$ and $L_R$. This does not penalize SoS preservation, as shown in Equation 2, which is equivalent to Equation 1 by dividing $m$ on both sides.

$$\sum_{S_i\in\mathcal{S}}\sum_{p_j^i\in S_i}\left(\frac{1}{\sqrt{m}}p_j^i\right)^2=n=\sum_{E_i\in\mathcal{E}}\sum_{e_j^i\in E_i}\left(\frac{1}{\sqrt{l}}\frac{e_j^i-\overline{e^i}}{\sigma_{e^i}}\right)^2. \tag{2}$$

By further analyzing the distributions of distances under an ideal setting, we find that scaling series by $\sqrt{1/m}$ and DEAs by $\sqrt{1/l}$ in $L_C$ and $L_R$ would largely stabilize their pairwise distance distributions. We formalize and prove this observation in Lemma 2.

*Lemma 2:* Given two series $S_1=\{p_1^1,\ldots,p_m^1\}$ and $S_2=\{p_1^2,\ldots,p_m^2\}$. Ideally, assume $p_1^1,\ldots,p_m^1,\ldots,p_1^2,\ldots,p_m^2$ are i.i.d. $\sim N(0,1)$. Scaling $S_1$ and $S_2$ by $\frac{1}{\sqrt{m}}$ will reduce the mean and variance of $d(S_1,S_2)$ by $\sqrt{m}$ and $m$, respectively.

*Proof:* As $\forall i\in[1,\ldots,m]$, $p_i^1$ and $p_i^2$ are i.i.d. $\sim N(0,1)$, we have

$$p_i^1-p_i^2\sim N(\mu_1-\mu_2,\sigma_1^2+\sigma_2^2)\sim N(0,2),$$
$$d(S_1,S_2)=\sqrt{\sum_{i=1}^{m}(p_i^1-p_i^2)^2}\sim\chi_m(\sqrt{2}),$$

where $\chi_m(\sqrt{2})$ is a centered $\chi$ distribution with $m$ degree of freedom, derived from i.i.d. variables $\sim N(0,2)$.

For $x\sim\chi_m(\sqrt{2})$, we have

$$\mu_{\chi_m(\sqrt{2})}=\sqrt{2}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})}\times\sqrt{2},$$
$$\sigma^2_{\chi_m(\sqrt{2})}=2\left(\frac{m}{2}-(\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})})^2\right)\times 2.$$

TABLE I
MEAN AND VARIANCE OF THE DISTRIBUTION OF PAIRWISE DISTANCES BETWEEN TWO IDEAL SERIES

| Length | Before Scaling | | $\times\sqrt{256/m}$ | | $\times\sqrt{1/m}$ | |
|---|---|---|---|---|---|---|
| $m$ | Mean | Var | Mean | Var | Mean | Var |
| 256 | 22.605 | 0.999 | 22.605 | 0.999 | 1.4128 | 0.0039 |
| 128 | 15.969 | 0.998 | 22.583 | 1.9961 | 1.4115 | 0.0078 |
| 96 | 13.820 | 0.997 | 22.569 | 2.6597 | 1.4105 | 0.0104 |
| 16 | 5.5692 | 0.984 | 22.277 | 15.743 | 1.3923 | 0.0615 |
| 8 | 3.8772 | 0.967 | 21.933 | 30.944 | 1.3708 | 0.1209 |

Scaling by $\sqrt{256/m}$ is to preserve SoS by scaling DEA itself. Scaling by $\sqrt{1/m}$ is the case where we scale both series and DEpA in loss functions. (Note that for length 256, scaling by $\sqrt{256}/m$ does not change the original behavior.)

Similarly, dividing $p_i^1$ and $p_i^2$ by $\sqrt{m}$, we have

$$\frac{p_i^1}{\sqrt{m}}\text{ and }\frac{p_i^2}{\sqrt{m}}\sim N\left(0,\frac{1}{m}\right),\ \frac{p_i^1}{\sqrt{m}}-\frac{p_i^2}{\sqrt{m}}\sim N\left(0,\frac{2}{m}\right),$$
$$d\left(\frac{S_1}{\sqrt{m}},\frac{S_2}{\sqrt{m}}\right)\sim\chi_m\left(\sqrt{\frac{2}{m}}\right).$$

And for $x\sim\chi_m\left(\sqrt{\frac{2}{m}}\right)$,

$$\mu_{\chi_m\left(\sqrt{\frac{2}{m}}\right)}=\sqrt{2}\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})}\times\sqrt{\frac{2}{m}},$$
$$\sigma^2_{\chi_m\left(\sqrt{\frac{2}{m}}\right)}=2\left(\frac{m}{2}-(\frac{\Gamma(\frac{m+1}{2})}{\Gamma(\frac{m}{2})})^2\right)\times\frac{2}{m}.$$

Thus, the conclusion holds that

$$\frac{\mu_{\chi_m(\sqrt{\frac{2}{m}})}}{\mu_{\chi_m(\sqrt{2})}}=\frac{1}{\sqrt{m}},\ \frac{\sigma^2_{\chi_m(\sqrt{\frac{2}{m}})}}{\sigma^2_{\chi_m(\sqrt{2})}}=\frac{1}{m}.$$

Typical examples of scaling are presented in Table I. We make three observations: (1) After scaling short series by $\sqrt{256/m}$, the means of distance distributions are comparable to series of length 256. This confirms that our design of SoS preservation is indeed helpful to preserve pairwise distances. (2) However, the variance of distance distributions increases dramatically after scaling, e.g., 16× from 0.984 to 15.743 for length 16. This introduces extra noise that hinders convergence. (3) By scaling both series and DEAs in the loss functions, not only are the means of distance distributions kept roughly the same (≈1.4), but also their variances are suppressed to a small value. This helps SEAnet focus on learning from the differences between series distance and DEA distance, without being interfered by endogenous noises of the distance distributions.

Finally, we observe that scaling series and DEA will not only keep the two distances to the same level, but will also largely stabilize the distance distributions. Both effects are beneficial to SEAnet's learning and convergence. Thus, by z-normalizing DEA, and scaling series and DEA in $L_C$ and $L_R$, SEAnet succeeds in providing high-quality DEAs by preserving SoS, and in converging fast to good optima (thanks to the stable latent variables and gradients).

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

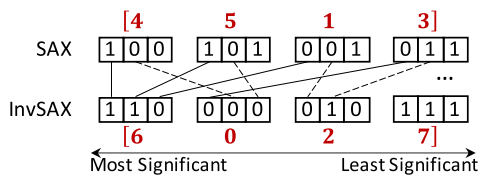


Fig. 6. SEAsam transformation and InvSAX [20].

### C. Sampling With SEAsam and SEAsamE

The representativeness of the training set upper bounds the quality of the deep models. To effectively train SEAnet on very large ($\geq 1e8$) series collections, a good sampling strategy is essential for providing representative subsets. This means that the sample should effectively cover the entire space of a given dataset, and we need to efficiently select this sample without having to perform expensive computations on the full dataset. For example, an effective but prohibitively expensive sampling strategy would be to sample from all leaves of an index built on the entire dataset, since such an index would cover the entire space, and each leaf would gather similar series.

To this end, we propose SEAsam (SEA-sampling), a novel data series sampling strategy based on the sortable data series representation, InvSAX [20]. Recall that SAX first transforms the data series into $l$ real values (i.e., the mean values of $l$ segments of consecutive points of the series), and then quantizes these real values, representing them using discrete symbols (usually of cardinality 256) [10]. The core observation is that every subsequent bit in a SAX word contains a decreasing amount of information about the location of its corresponding data point, and simply increases the degree of precision. Interleaving SAX's bits such that all significant bits across each SAX word precede all less significant bits presents a value array with descending significance, i.e., InvSAX.

The procedure to generate InvSAX is shown in Fig. 6. The most significant bits {1,1,0,0} across the original SAX words are moved to the first 4 bits of InvSAX, making its first and most significant value 6 (shown in red/bold). The second most significant bits {1,0,1,1} are moved to InvSAX's 5-8 bits, making the second value 2. The last bits {0,1,1,0} are moved to InvSAX's 9-12 bits, making the last two values 2 and 6. Thus, this series will be order by its InvSAX [6,2,2,6].

As illustrated in Fig. 7, SEAsam orders the series collection by their InvSax representations, and draws samples at equal-intervals (e.g., every 1,000 series) from this sorted order. Thus, SEAsam samples are expected to preserve the distribution of the series collection by evenly covering its InvSAX space. Moreover, the time complexity of SEAsam is $\mathcal{O}(nm)$, and the space complexity of SEAsam is $\mathcal{O}(nl)$, rendering SEAsam an efficient strategy.

Algorithm 1 provides the pseudo-code of SEAsam sampling. Line 1 generates SAXs for series collection $\mathcal{S}$. Lines 3-8 interleave SAX bits to generate InvSAX. This transformation could be implemented in place if under memory pressure, i.e., replacing SAX by its corresponding InvSAX. Lines 9-10 sort InvSAX, select $n'$ equal-interval indices and return the corresponding sampled series. After sampling from the data series collection, we randomly permute the samples to make training and validation sets during each epoch of autoencoder training.

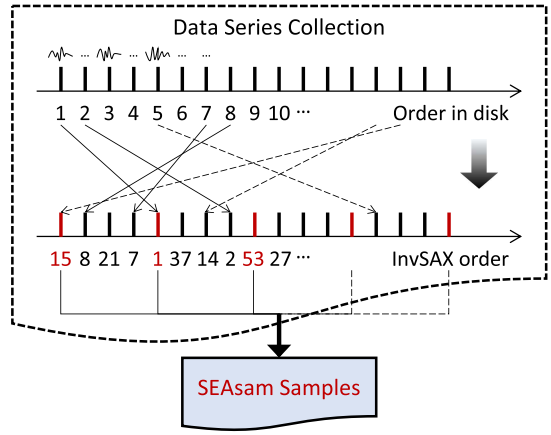


Fig. 7. SEAsam sampling workflow.

**Algorithm 1:** SEAsam Sampling.

**Input:** data series collection $\mathcal{S}$ of size $n$, sample size $n'$
**Output**: sample set $\mathcal{S}'$
1: $\hat{\mathcal{S}} = \text{SAXTransform}\mathcal{S}$
2: $\text{Inv}\hat{\mathcal{S}} = \emptyset$
3: **for all** $\hat{S} \in \hat{\mathcal{S}}$ **do**
4: $\quad \text{Inv}\hat{S} = 0_{SAX}$
5: $\quad$ **for all** $i \in$ RANGE(SAX cardinality) **do**
6: $\quad\quad$ **for all** SAX symbol $\hat{s} \in \hat{S}$ **do**
7: $\quad\quad\quad \text{Inv}\hat{S}$.append(the $i$ bit of $\hat{s}$)
8: $\quad \text{Inv}\hat{\mathcal{S}}$.add($\text{Inv}\hat{S}$)
9: indices $=$ SORT($\text{Inv}\hat{\mathcal{S}}$).indices[: : $n/n'$]
10: $\mathcal{S}' = \mathcal{S}$[indices]

*Extending SEAsam With SEAsamE:* The empirical effectiveness of SEAsam sampling (cf. Section 9) and the importance of the training set quality for our problem motivate the need to develop better and tailored sampling strategies. Designing sampling strategies demands a close investigation into the sampling space. Although SEAsam sampling is designed to represent well the raw data series space, a key observation is that there are more spaces to be explored for DEA learning. We first provide a discussion of the different sampling spaces in DEA learning. On this basis, a novel combined sampling strategy, SEAsamE, is proposed in order to provide a combined use for all these spaces.

Our main observation is that there are three major sampling spaces in DEA learning, which could be organized into two categories.

The first category regards the dataset-specific sampling spaces, including the raw data series space targeted by SEAsam. Another important sampling space falling into this category is the distance distribution of all possible series pairs in the collection, which is exactly what we would like SEAnet to preserve in the lower-dimensional DEA space. From the perspective of deep model training, these two spaces correspond to the distributions of raw features and targeted values. Preserving both distributions has been deemed beneficial in previous studies for capturing the ground truth and avoid bias in deep learning [45].

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

**Algorithm 2:** Training With SEAsamE.

**Input**: deep model with encoder $\phi$ and decoder $\psi$ parameterized by $\Theta$, data series collection $\mathcal{S}$, training size $n'$ and candidate size $n'_c$, batch size $b_s$ for series and $b_p$ for pairs, max epochs max$e$
**Output**: learned parameters $\Theta$
1: estimate $\bar{P}(d)$
2: **for all** $i \in$ RANGE(max$e$) **do**
3: **if** SAMPLECHECK($i, \Theta$) **then**
4: $\mathcal{S}' =$ SAMPLESERIES($\mathcal{S}$, $\Theta$, $n'$, $n'_c$)
5: **for all** mini-bach $\mathcal{S}'_b \in \mathcal{S}'$ **do**
6: $E'_b = \phi(\mathcal{S}'_b)$, $\mathcal{R}'_b = \psi(E'_b)$
7: $\mathcal{S}'_p =$ SAMPLEPAIRS($\mathcal{S}'_b$, $\bar{P}(d)$, $b_p$)
8: $L = L_C(\mathcal{S}'_p, E'_p) + \alpha L_R(\mathcal{S}'_b, \mathcal{R}'_b)$
9: backpropagate $\nabla L$

**Function** SAMPLESERIES$\mathcal{S}$, $\Theta$, $n'$, $n'_c$
10: $\mathcal{S}'_c =$ SEASAM($\mathcal{S}$, $n'_c$)
11: $\mathcal{R}'_c = \psi \cdot \phi(\mathcal{S}'_c)$
12: estimate $\bar{P}(L_R)$ according to $L_R(\mathcal{S}'_c, \mathcal{R}'_c)$
13: draw $\mathcal{S}'$ of $n'$ series $\sim \bar{P}(L_R)$
14: **return** $\mathcal{S}'$

**Function** SAMPLEPAIRS($\mathcal{S}'_b$, $P(d)$, $b_p$)
15: calculate $d(\mathcal{S}'_b, \mathcal{S}'_b)$
16: draw $\mathcal{S}'_p = \{(\mathcal{S}'_i, \mathcal{S}'_j)\}$ of $b_p$ pairs $\sim \bar{P}(d)$
17: **return** $\mathcal{S}'_p$

The second category regards the model-specific sampling spaces. Intuitively speaking, these spaces represent model's response to the input features, i.e., the difficulty to embed and reconstruct a series. In DEA learning, the reconstruction errors serve this purpose. Unlike preserving the same distributions for the dataset-specific sampling spaces, different strategies facilitate model training following different heuristics [46], e.g., sampling harder series is expected to offer faster convergence. In this work, we propose to draw samples evenly in terms of their reconstruction error values (instead of reconstruction error frequencies). That is, the sample set has a uniform distribution of reconstruction errors. This sampling strategy is capable to correct the bias while controlling the gradient variance [46].

In order to provide a training set that represents all three major sampling spaces mentioned above, we propose a novel composite sampling strategy SEAsamE. SEAsamE incorporates SEAsam to preserve the raw series space, with extra steps for the other two spaces. An extra preprocessing step is introduced to collect an empirical distribution of the pairwise distances $\bar{P}(d(S_i, S_j))$. To draw a set of training series, SEAsamE first draws a candidate set of SEAsam samples. Second, reconstruction errors $L_R(\mathcal{S})$ of the candidate series are calculated. The training set is drawn based on the inverse of the frequencies of reconstruction errors $1/\bar{P}(L_R(S_i))$. To draw series pairs for the calculation and backpropagation of $L_C(\mathcal{S}, \mathcal{S})$, distances of all series pairs within each mini-batch are calculated. Then, SEAsamE selects series pairs according to their estimated distance probabilities $\bar{P}(d(S_i, S_j))$.

Algorithm 2 shows how to train SEAnet with SEAsamE. Line 1 is a preprocessing step to estimate $\bar{P}(d)$. (This step can be implemented by calculating histograms based on a sufficient number of random samples.) At the beginning of each epoch, line 3 checks whether to invoke function SAMPLESERIES, and then line 4 draws a new series sample set. SAMPLESERIES draws $n'_c$ SEAsam samples $\mathcal{S}'_c$ in line 10, calculates their reconstruction errors in line 11-12, and returns $n'$ samples from $\mathcal{S}'_c$ according to $\bar{P}(L_R)$. Lines 5-6 consider each mini-batch and calculate their DEAs and reconstructions. To draw $b_p$ series pairs, line 7 invokes function SAMPLEPAIRS, which calculates distances for all series pairs in line 15, and returns $b_p$ sampled pairs according to $\bar{P}(d)$. Last, lines 8-9 calculate and backpropagate the loss $L$.

## V. Experimental Evaluation

We present our experimental evaluation of SEAnet, DEA-based data series similarity search, SEAsam and SEAsamE using 7 diverse synthetic and real datasets. In summary, the results demonstrate that the SEAnet DEA is robust across various dataset properties and outperforms its competitors by better preserving original pairwise distances and nearest neighborhood structure, leading to better approximate similarity search results than traditional (PAA-based) and alternative deep learning (DEA-based using FDJNet, TimeNet, and InceptionTime) approaches.

*Setup:* All deep models were trained using Nvidia Tesla V100 SXM2 (16G memory). Sampling, indexing and query answering were conducted in a server with 2x Intel(R) Xeon(R) Gold 6134 CPU @ 3.20 GHz and 320 GB RAM. Software environments were python/3.6.10, pytorch-gpu/py3/1.5.1 and cuda/10.2.

*Datasets:* Experiments were conducted on 3 synthetic datasets of different characteristics and 4 real datasets from diverse domains. For synthetic datasets, we used RandWalk, F5 and F10. RandWalk [2], [8] was generated as cumulative sums of steps following a standard Gaussian distribution $N(0, 1)$. F5 and F10 were recently introduced to evaluate iSAX on datasets of different frequencies [12]. They were generated through Inverse Discrete Fourier Transform (IDFT) from a random spectrum with its first 5 or 10 components being amplified. The 4 real datasets are Seismic from seismology, Astro from astronomy, SALD from neuroscience and Deep1B from image processing [2]. Length of each series is 128 for SALD, 96 for Deep1B, and 256 for the rest. Note that these datasets are considered hard for similarity search [2], [8]. We experimented with dataset sizes between 1M to 100M series (100M series of length 256 $\approx$100 GB).

*Methods:* We evaluated the SEAnet-generated DEA and its applications in data series similarity search against PAA and DEA generated by SEAnet-nD (a simplified version of SEAnet), and our adaptations of FDJNet [18], TimeNet [21], and InceptionTime [3]. We describe these methods below. SEAsam was compared to uniformly random sampling.

PAA is the baseline method to evaluate DEA's summarization quality. PAA-based MESSI iSAX index [11] is the SOTA baseline method to evaluate DEA-based iSAX on approximate similarity search. When it is clear from the context, we used the term PAA to denote both the PAA summarization, and the PAA-based iSAX index in the rest of Section V. We used the same convention for other methods, as well, e.g., SEAnet

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

TABLE II
SEANET ENCODER ARCHITECTURE SPECIFICS FOR DIFFERENT SERIES LENGTHS

| layer name | 13-layer (*96*) | 15-layer (*128*) | 17-layer (*256*) |
|---|---|---|---|
| ConvLayer | $[3, 96]$ | $[3, 128]$ | $[3, 256]$ |
| ConvBlock | $\begin{bmatrix} 3, 96 \\ 3, 96 \end{bmatrix} \times 5$ | $\begin{bmatrix} 3, 128 \\ 3, 128 \end{bmatrix} \times 6$ | $\begin{bmatrix} 3, 256 \\ 3, 256 \end{bmatrix} \times 7$ |
| Linear | $\begin{bmatrix} 96 \\ 16 \end{bmatrix}$ | $\begin{bmatrix} 128 \\ 16 \end{bmatrix}$ | $\begin{bmatrix} 256 \\ 16 \end{bmatrix}$ |
| parameters/M | 0.585 | 1.234 | 5.712 |
| MACs/B | 0.053 | 0.152 | 1.412 |

Building blocks ([kernel size, channel number] for the convolutional layer/block, [dimension] for the linear layer) are shown in brackets, multiplied by the numbers of blocks stacked.

denotes the DEA generated by SEAnet and the index built on the DEA generated by SEAnet.

SEAnet-nD is an encoder-only version of SEAnet, introduced to evaluate the contribution of the decoder to the final performance. The convolution kernel size for SEAnet and SEAnet-nD is 3. For FDJNet, we adopted the same network setting as SEAnet-nD. For TimeNet, we used the output of the last position as DEA, instead of the original concatenation of latent vectors. This enables TimeNet using longer latent vectors to generate lower-dimensional DEA. The dropout probability of TimeNet is 0.4, following the original setting [21]. For InceptionTime, we used the same structure as SEAnet, but replaced ResBlock with InceptionBlock [3]. The convolution kernel sizes for InceptionTime are {3, 5, 9, 17}. Batch size was set to 128 for TimeNet (due to our memory limit), and to 256 for the other architectures. For all architectures, we stacked 7, 6, and 5 building blocks for series of length 256, 128, and 96, respectively. Latent dimensions and channels were the same to series length. All models were trained using SGD and the same loss function (cf. Section IV-A). Training size was 200,000 series, and validation size was 20,000. TimeNet was trained for 125 epochs, while the others for 100 epochs. Hyperparameters were tuned for each model (of specific DEA length) on 100M datasets. The best hyperparameters for similarity search were adopted for all other dataset sizes. $\alpha$ was searched from {0.1, 0.25, 0.5, 1, 1.25}. Learning rate was cross searched from {$1e$-3, $5e$-3, $1e$-2, $2.5e$-2, $5e$-2}, and was either linearly decayed (every epoch), or exponentially decayed (by 0.9 every 2 epochs) until $1e$-5. Totally, 6,090 deep models were trained to provide a thorough profile of DEA architectures. Other hyperparameters were set to their default values. For indexing, leaf size $h$ was 8,000 by default.

SEAnet encoder architecture details are reported in Table II. We use the respective symmetrical architectures as their decoders. The complexities were measured in number of (millions of) tunable parameters, and (billions of) Multiply-and-Accumulate (MAC) operations. All models converged with the training speed reported in Fig. 17.

*Measures:* To evaluate summarization quality, we used three measures: average distance differences (unscaled $L_C$), reconstruction RMS, and NN coverage. Series subsets, or series pairs are SEAsam samples from 100M datasets.

1) *Average Distance Differences ($L_C$):* Differences between original distances and DEA distances of series pairs, i.e., $|d'(\phi(S_i), \phi(S_j)) - d(S_i, S_j)|$. Reported values were averaged from 20,000 pairs. (Differences of 1,000 pairs were illustrated as scatters in Fig. 11.)
2) *Reconstruction RMS:* Root-Mean-Square errors between original series and their reconstructions, i.e., $\sqrt{\frac{1}{m}\sum_i (p_i - p'_i)^2}$, where $S' = [\ldots, p'_i, \ldots]$ is the reconstruction of series $S = [\ldots, p_i, \ldots]$. Reported values were averaged from 20,000 series.
3) *NN Coverage:* The coverage of series $S$'s nearest neighbors in DEA space, i.e., $\frac{|k\text{NN}_d(S) \cap k\text{NN}_{d'}(E)|}{|k\text{NN}_d(S)|}$, where $k\text{NN}_d$ and $k\text{NN}_{d'}$ return $k$ nearest neighbors in original and DEA spaces respectively. We consider NN coverage as a direct measure of whether the structure of original distance spaces is preserved or not. We report NN coverage for $k \in \{1, 5, 10, 50, 100, 500, 1{,}000\}$ in Section V-B. The reported values were averaged from 1,000 series, whose kNN was searched from 20,000 series.
4) *1st BSF Tightness:* To evaluate the DEA performance on data series similarity search, we used the tightness of the first Best-So-Far (1st BSF) [2]. In the context of approximate similarity search, 1st BSF is the best result under the constraint of a fixed number of leaf nodes, or series allowed to be examined by the query answering algorithm. In the case where only one leaf node is allowed to be examined, the 1st BSF is also called the *approximate answer*. In our experiments (Sections V-A and V-C), we report the 1st BSF tightness as a function of the number of series examined (this makes for a fair comparison across indices with leaves containing different number of series). Similar to previous work [2], [8], we report the average tightness over 1,000 queries.

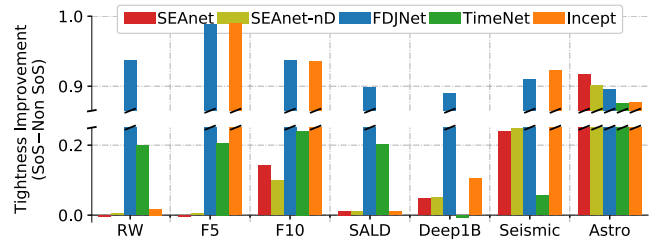


Fig. 8. Improvements of employing SoS preservation in terms of 1st BSF tightness (positive values mean SoS is better); 100M datasets.

### A. SoS Preservation and SEAsam

In this section, we evaluate the two novel design choices we propose for DEA methods, i.e., the SoS preservation principle and SEAsam.

*SoS Preservation:* First, we evaluate the effect of the scaling steps introduced by the SoS preservation. Fig. 8 summarizes the results. We trained all five models using SEAsam samples across seven 100M-size datasets and reported their 1st BSF tightness improvements. 1st BSFs were reported under the constraint that the query answering algorithm examines a maximum of 10,000 series in the index before producing the answer. Improvements

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

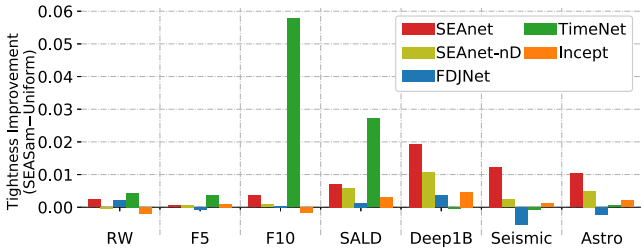


Fig. 9. Improvements of SEAsam over uniformly random sampling in terms of 1st BSF rightness (positive values mean SEAsam is better); 100M datasets.

are calculated by subtracting the 1st BSF tightness of the non-scaled models from that of the scaled models.

The results show that the scaled models provided better 1st BSF in 32 out of the 35 experiments (91%). The only three exceptions were by small margins. Besides, 14 of 35 (40%) non-scaled models could not effectively converge. They either converged to bad local optima of large constant DEAs (cf. Section IV-B), or did not converge and generate random DEAs (exhibiting similar statistics to Table I).

These results verify that the proposed SoS preservation is indeed an effective method for both preserving pairwise distances and facilitating network convergence. We note that the SoS preservation idea is applicable to any suitable architecture, and apart from SEAnet, it also improves the performance of the non-scaled versions of FDJNet, TimeNet and InceptionTime. In the rest of this section, we only report results using the scaled models.

*SEAsam:* Second, we compare the proposed SEAsam against the commonly used uniform random sampling in Fig. 9. Results are reported similarly to the previous experiments. Improvements were calculated by subtracting the 1st BSF tightness of models trained using random samples from those trained using SEAsam samples.

For 27 out of the 35 experiments (77%), SEAsam provided tighter 1st BSFs than random sampling. SEAsam was only surpassed by random sampling on 8 experiments (23%) with a very small margin. We observe that for SEAnet, SEAsam was always better.

In order to evaluate the effectiveness of SEAsam, we also measured the number of distinct leaves (of an index constructed on the entire dataset) containing a series that is part of the SEAsam sample in Fig. 10. Intuitively, the leaf nodes of the index represent an effective split of the series space, which corresponds to the underlying distribution of the collection. The more leaf nodes a sample set covers, the better it represents the entire collection. In our experiments, for all samples with sizes between 10K-500K series across our 7 datasets, SEAsam samples covered more leaf nodes than uniformly random samples with an average improvement of 8%, and up to 28% for the challenging F10 dataset.

These results verify that SEAsam provides more representative samples than uniformly random sampling. In the following, we report results with SEAsam.

### B. DEA Quality

*Average Distance Differences:* The averaged distance differences reported in Table III(a), show that SEAnet and SEAnet-nD outperformed PAA in all 7 datasets. SEAnet and SEAnet-nD also outperformed all other architectures in 6 out of the 7 datasets. The averaged distance differences for SEAnet were better than SEAnet-nD for the 3 synthetic datasets, but worse for the 4 real datasets. This is because of the regularization effect of the decoder. Synthetic datasets are less noisy, making models prone to overfitting to the training sets. In this case, the decoder's regularization improves the averaged distance differences. On the other hand, real datasets are more noisy, where loss $L_R$ dominates $L_C$, making SEAnet worse than SEAnet-nD in terms of averaged distance differences. However, as we will explain later on, SEAnet still outperformed SEAnet-nD in terms of NN coverages and 1st BSF tightness.

Regarding the other models, TimeNet worked better than FDJNet and InceptionTime only for the F5 dataset, which is of moderate periodicity, and lagged behind for Deep1B, whose adjacent values are less correlated. InceptionTime's high performance on Astro is an interesting result. However, after examining the embedding and reconstructed series of InceptionTime on Astro, we infer this is due to overfitting (reconstruction RMS, NN coverage, 1st BSF tightness and other results concur to this explanation).

*Distance Scatter:* Distance differences are depicted in scatter plots in Fig. 11. Points close to the $y = x$ diagonal correspond to series for which the original distances are well preserved in the DEA space. We observe that scatters of DEAs generated by SEAnet assembled tighter than PAA around the diagonal for all 7 datasets. Moreover, scatters of DEAs exhibited stronger linearity; thus, SEAnet preserved the true nearest neighborhoods better than PAA.

*Reconstruction RMS:* The reconstruction RMS results are reported in Table III(b). SEAnet surpassed competitors on 5 out of 7 datasets; it lost to PAA for Seismic and Astro. Upon close examination, we observe that this happened because of some hard to summarize series, for which neither SEAnet, nor PAA succeeded to produce a good summarization.

Given that the data series are z-normalized, an RMS $\approx$1 might imply useless local optima reached by setting all reconstruction values to zeroes. This is exactly the case for TimeNet on Deep1B/Astro. Such decoders cannot contribute at all to better summarizations. This is even worse than getting higher RMS, where the network might still learn from data. Such observations prove that SEAnet is more robust than PAA, FDJNet, TimeNet and InceptionTime for datasets of different characteristics.

*NN Coverages:* The NN coverages are reported in Fig. 12. SEAnet outperformed PAA and other architectures on all 63 experiments. This observation confirms SEAnet's capabilities on well-preserving original distance space structures in the DEA space. The advantage of deep models over PAA is not as obvious as in Table III(a), except for SEAnet, indicating that the target of preserving original distances in DEA distances alone cannot guarantee to provide high-quality DEAs for similarity

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

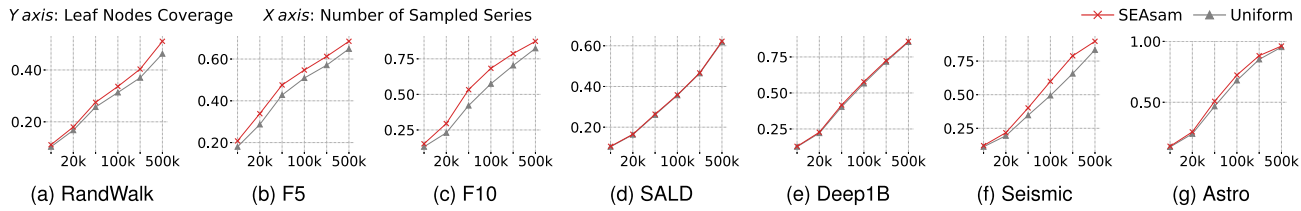


Fig. 10. Leaf node coverage across different sampling sizes for uniformly random sampling (light gray) and SEAsam (dark red).

TABLE III
(a) AVERAGED DISTANCE DIFFERENCES BETWEEN PAIRS OF SERIES IN THE ORIGINAL AND THE EMBEDDED (PAA, DEA) SPACES. (b) ROOT-MEAN-SQUARE ERROR BETWEEN ORIGINAL AND RECONSTRUCTED SERIES. IN BOTH CASES, BEST RESULT (LOWER IS BETTER) IS MARKED IN BOLD, SECOND BEST IS UNDERLINED

| | (a) Averaged Distance Differences | | | | | | (b) Reconstruction RMS Error | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | PAA | FDJNet | TimeNet | Incept | SEAnet-nD | SEAnet | PAA | TimeNet | Incept | SEAnet |
| RandWalk | 1.3701 | 0.2794 | 0.4098 | 0.6285 | 0.2976 | **0.2194** | 0.3061 | 0.3354 | 0.3587 | **0.2604** |
| F5 | 2.1152 | 0.1737 | 0.2103 | 0.2836 | 0.1692 | **0.1629** | 0.4214 | 0.2527 | 0.2708 | **0.2433** |
| F10 | 5.0395 | 1.1943 | 1.9063 | 1.3958 | 1.1859 | **1.1672** | 0.6238 | 0.6799 | 0.5041 | **0.2635** |
| SALD | 3.2927 | 0.6247 | 0.6928 | 0.858 | **0.5748** | 0.6182 | 0.5586 | 0.5883 | 0.6831 | **0.5023** |
| Deep1B | 8.1095 | 0.9511 | 7.8478 | 0.9511 | **0.9083** | 0.9484 | 0.9207 | 1.0 | 0.6368 | **0.5418** |
| Seismic | 9.9629 | 1.3798 | 1.6577 | 1.4555 | **1.306** | 1.4514 | **0.7385** | 0.8211 | 0.9669 | 0.7771 |
| Astro | 14.622 | 1.9239 | 2.4981 | **1.7983** | 1.8991 | 1.9737 | **0.9267** | 1.0 | 1.4096 | 1.1196 |

Results calculated using 10,000 series seasam sampled from datasets of 10-million series.

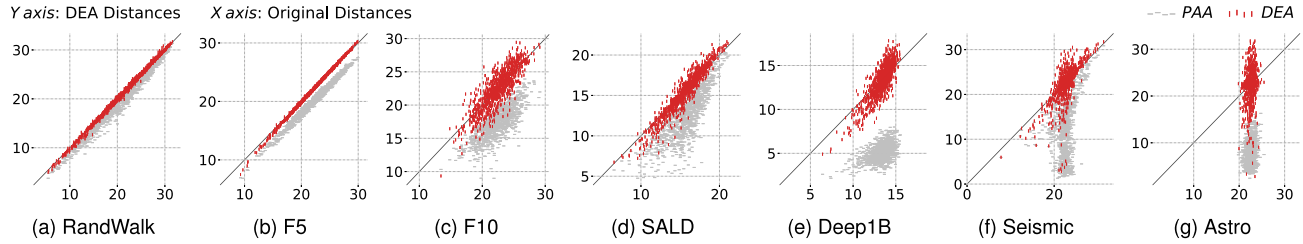


Fig. 11. Distance scatters of 1,000 SEAsam sampled series pairs across different datasets for PAA (light gray) and SEAnet (dark red).

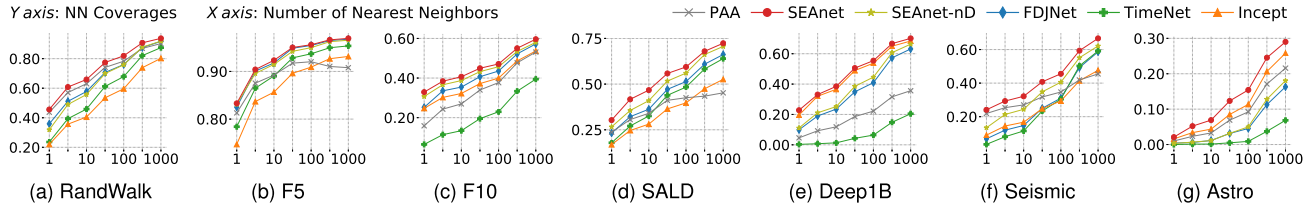


Fig. 12. Nearest neighbors' coverage versus neighborhood size (higher is better).

search. This observation, together with the fact that SEAnet outperformed SEAnet-nD, confirms the need for the decoder.

SEAnet-nD outperformed FDJNet on 56 out of the 63 experiments (89%). This confirms the overall SEAnet design choices over FDJNet, even after FDJNet was improved by using SoS preservation. There were clear gaps on Deep1B in Fig. 12(e) between convolutional models and PAA, and between PAA and TimeNet. This is because Deep1B is from image processing, where adjacent values are not necessarily correlated. This once again attests to SEAnet's versatility in handling datasets with different properties.

### C. DEA for Approximate Search

*Approximate Answers Tightness:* Fig. 13 reports the results for the evaluation of the quality of the approximate similarity search answers. That is, we evaluate the quality of the 1st BSF answers when the search algorithm examines a single leaf node of the index. We note that although this kind of evaluation is used in the literature [2], [10], it is not a fair comparison, as the visited leaf sizes may vary across different experiments, datasets, and indexes, depending on the distribution properties of the DEA values. If the DEA distribution is very skewed (e.g., with many

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

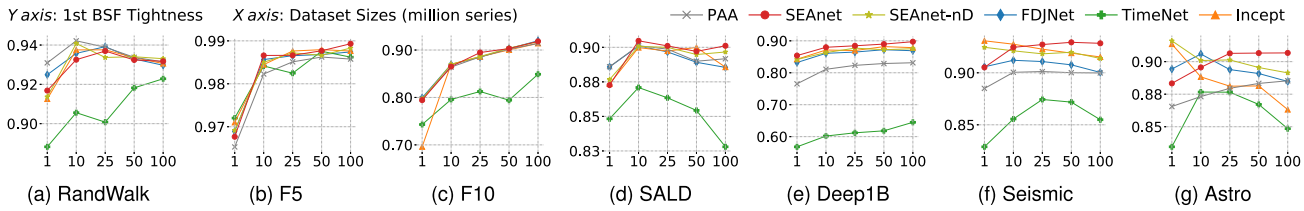


Fig. 13. Approximate query answers quality: 1st BSF tightness versus dataset size (higher is better).

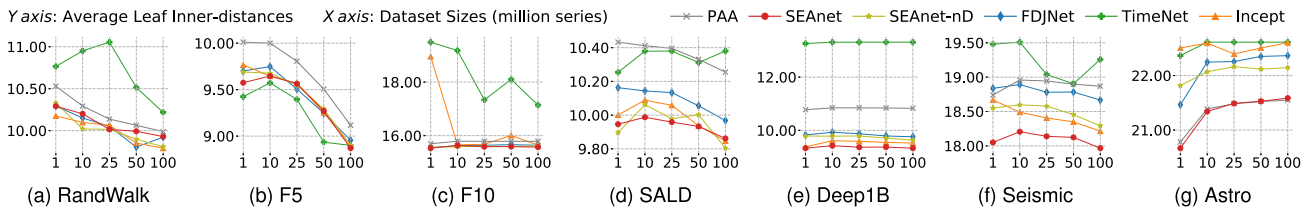


Fig. 14. Average distances among series of the visited leaf nodes versus dataset size (lower is better).

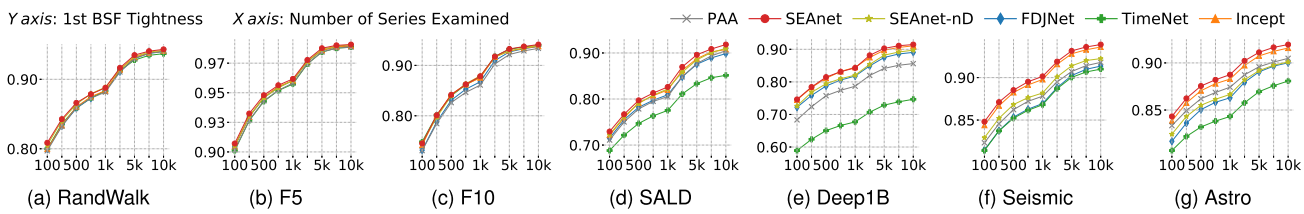


Fig. 15. Approximate query answers quality: 1st BSF tightness versus number of series visited; 100M series datasets (higher is better).

DEAs having the same or similar values), then they tend to gather into a small subset of the leaf nodes, resulting to a few large leaf nodes. This has two consequences:

1) Visiting a single large leaf involves examining a large number of series, which implies an increased probability for finding a tighter 1st BSF. Nevertheless, this problem is alleviated as the dataset size increases (e.g., when we move from the 1M to the 100M RandWalk dataset, the additional number of series visited by PAA-based iSAX, when compared to SEAnet DEA-based iSAX, drops from 205% to 7%).
2) If the produced DEAs are very similar to one another for a large dataset, then the index cannot even be built. This happens when a leaf node has exhaustively used all iSAX bits and cannot be split any further in order to accommodate the incoming series. This situation generally occurred for DEAs originating from the non-scaled models, and for PAA on the 100M SALD dataset. SEAnet avoids this problem across all our experiments.

Benefiting from leaf nodes of larger sizes, PAA or other models outperform SEAnet in some of the 1M and 10M dataset experiments, even though their summarization qualities are not better (refer to Table III(a) and Fig. 12). SEAnet and SEAnet-nD show their superiority in the large (≥25M) and hard (real) datasets. TimeNet generally lags behind the other solutions, indicating that a direct application of recurrent models is less competitive than convolutional models. This could come from the fact that RNN models are designed to capture temporal correlations while being relatively less powerful at extracting local patterns [16].

*Leaf Node Compactness:* To better understand the quality of DEA-based iSAX, we examine the compactness, i.e., the average distances among series of the visited leaf nodes in Fig. 14. Smaller average distance indicates the index is better in terms of mapping similar series into the same leaf nodes. In RandWalk and F5, the advantages of SEAnet are not obvious as all methods perform well. However, when considering the harder datasets, SEAnet achieves better results in 22 out of the 25 experiments. These results demonstrate that SEAnet produces more compact indexes than the competitors, especially on datasets considered hard for similarity search.

*1st BSF Tightness Limited by Number of Series to Examine:* We evaluate the benefit of using DEA for similarity search, by reporting the 1st BSF tightness as a function of the number of series that the similarity search algorithm examines. The results on 100M datasets, are shown in Fig. 15. SEAnet improved the 1st BSF tightness, and thus the similarity search results, in 61 out of the 63 experiments. Its advantage was particularly obvious on the Deep1B, Seismic, and Astro (hard) datasets.

Besides the 1st BSF tightness, the index's leaf node compactness (i.e., the average distances among the series in a leaf node) also profiles the quality of the index built on DEA or PAA. Smaller average distance indicates the index is more successful in grouping similar series into the same leaf node.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

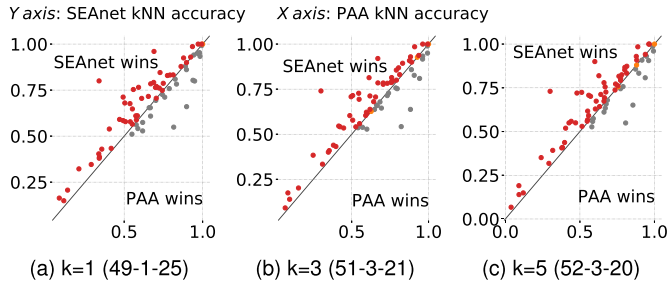


Fig. 16. SEAnet DEA versus PAA on kNN classification accuracies of UCR18 dataset. Number of datasets where SEAnet wins-ties-losses compared with PAA are reported.

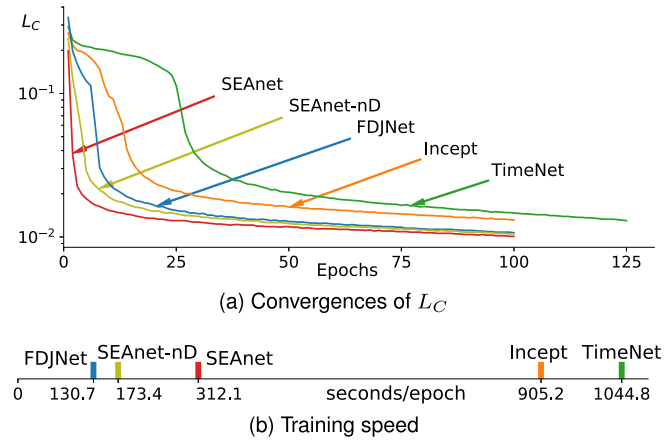


Fig. 17. Convergence procedures measured by compression error (averaged distance difference) $L_C$ across architectures on F5 dataset.

SEAnet leaded to an average improvement over PAA in leaf node compactness of 4%, and up to 14% for the challenging real dataset Deep1B, demonstrating its effectiveness in producing more compact indexes than the SOTA competitors.

To conclude this section, our comprehensive experiments verify the effectiveness of SEAnet's ability to provide better DEAs to facilitate data series similarity search.

### D. DEA for Downstream applications

*SEAnet for Classification:* We now evaluate the utility of SEAnet DEA in other (than similarity search) downstream applications. Fig. 16 reports the DEA/PAA-based $k$-NN classification accuracy on the UCR time series classification archive [47] (we used the 75 datasets with $\geq$100 equal-length series and no missing values). With $k$=1, 2, 3, SEAnet DEAs outperformed PAA on 65%, 68% and 69% of the datasets respectively, demonstrating the superiority of SEAnet DEA over PAA for classification tasks, as well.

### E. Time and Convergence

*Training Time:* We report training times and convergence for all architectures in Fig. 17. (We omit query answering times because they are proportional to the number of visited series and hence similar among different methods.) Benefiting from the encoder-only architecture, FDJNet and SEAnet-nD the fastest models to converge in Fig. 17(b). For architectures with both encoder and decoder, SEAnet is 2.9$X$ and 3.35$X$ faster than InceptionTime and TimeNet under similar parameters. SEAnet converges more steadily to the lowest training loss (Fig. 17(a)).

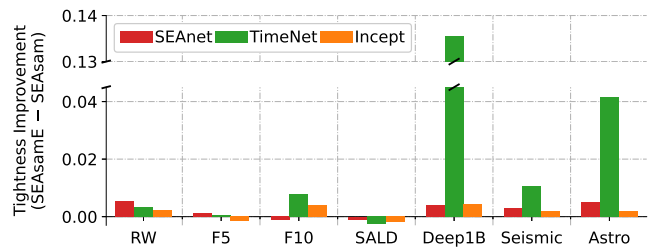


Fig. 18. Improvements of SEAsamE over SEAsam in terms of 1st BSF rightness (positive values mean SEAsamE is better); 100M datasets.

### F. SEAsamE

*SEAsamE for Similarity Search:* We evaluate the effectiveness of training SEAnet with SEAsamE for similarity search, by reporting the 1st BSF tightness improvements when compared to SEAsam. The 1st BSFs are reported under the constraint that the query answering algorithm examines a maximum of 10,000 series in the index before producing the answer. The results on 100M datasets, are shown in Fig. 18. For 16 out of the 31 experiments, SEAsamE provided tighter 1st BSFs than SEAsam. Its advantage was especially obvious for TimeNet, demonstrating that RNN models can also be promising solutions when combined with specialized sampling and training strategies.

### G. SEAtrans

*SEAtrans for Tightness Upper Bounds:* To further analyze the improvement space of DEA-based data series similarity search, we reported the 1st BSF tightness upper bounds of deploying different summarizations in Fig. 19. The 1st BSF tightness *upper bounds* were extracted by examining series according to their distances to the query series in the summarization space. These ideal results reveal the importance of considering the optimization opportunities deriving from both the index structure *and* the summarizations.

Fig. 19 demonstrates that SEAnet significantly improves the 1st BSF tightness upper bounds achieved by PAA. Moreover, SEAtrans further improves these bounds for four hard real-world datasets, where this is more needed. Compared with Fig. 15, these results lead to the following three main observations. (1) For RandWalk, F5 and F10, all summarizations provided almost perfect upper bounds ($\approx$1). Hence, the index structure should be better tuned to approach the capabilities of summarizations. (2) For SALD and Deep1B, DEA learned by SEAtrans provided better upper bounds ($\approx$1) than PAA ($<$1). In such cases, tuning the index structure is more promising for improving the 1st BSF tightness than tuning the DEA qualities. (3) For Seismic and Astro, DEA learned by SEAtrans provided better upper bounds ($\approx$96–98%) than the competitors, but with room for

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

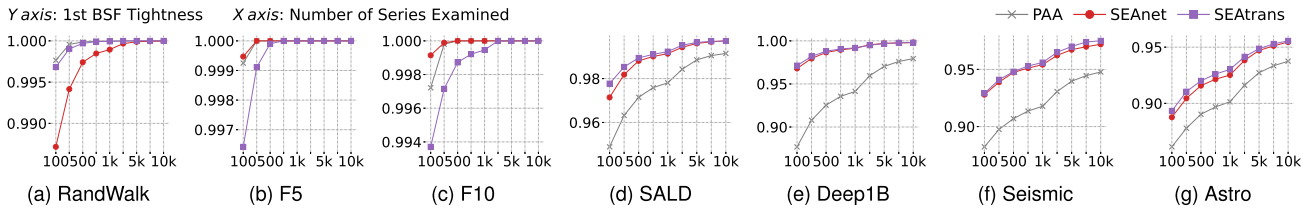


Fig. 19. Query answers quality upper-bounds: 1st BSF tightness versus number of series visited; 100M series datasets (higher is better).

improvement. This implies that improving the DEA quality on these datasets is crucial in order to further improve similarity search. These observations are important guidelines for future studies in this direction.

## VI. Discussion and Conclusion

In this article, we introduce the use of deep learning embeddings, DEA, for data series similarity search. We propose novel autoencoders, SEAnet and SEAtrans, designed under the SoS preservation principle, for effectively learning DEA. New sampling strategies, SEAsam and SEAsamE, are introduced in order to facilitate training on massive collections. We demonstrate that the DEA learned by SEAnet and SEAtrans more closely approximates the original data series distances, better preserves the true nearest neighbors in the summarized space, better reconstructs the original series, and leads to better similarity search results than the SOTA PAA-based iSAX. These preliminary results are very promising, they set the ground for further advancements in this area, and have the potential to also improve the performance of kNN classification, anomaly detection and other similarity search based applications.

In our future work, we will study the development of lower bounding properties for DEA that will enable exact similarity search, the adaptation of transfer learning or incremental learning techniques for quickly fitting new or dynamic datasets, the development of more powerful sampling strategies, and the careful study of query answering strategies on top of DEA, including *product quantization* [48], *locality sensitive hashing* [49], and modern series indexes [30].

## References

[1] T. Palpanas and V. Beckmann, "Report on the first and second interdisciplinary time series analysis workshop (ITISA)," *ACM SIGMOD Rec.*, vol. 48, pp. 36–40, 2019.

[2] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "Return of the lernaean hydra: Experimental evaluation of data series approximate similarity search," *Proc. VLDB Endowment*, vol. 13, pp. 403–420, 2019.

[3] H. I. Fawaz et al., "Inceptiontime: Finding alexnet for time series classification," *Data Mining Knowl. Discov.*, vol. 34, pp. 1936–1962, 2020.

[4] J. Paparrizos and L. Gravano, "k-Shape: Efficient and accurate clustering of time series," *ACM SIGMOD Rec.*, vol. 45, pp. 69–76, 2016.

[5] P. Boniol, J. Paparrizos, T. Palpanas, and M. J. Franklin, "SAND: Streaming subsequence anomaly detection," *Proc. VLDB Endowment*, vol. 14, pp. 1717–1729, 2021.

[6] A. Gogolou, T. Tsandilas, T. Palpanas, and A. Bezerianos, "Comparing similarity perception in time series visualizations," *IEEE Trans. Vis. Comput. Graph.*, vol. 25, no. 1, pp. 523–533, Jan. 2019.

[7] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. J. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining Knowl. Discov.*, vol. 26, pp. 275–309, 2013.

[8] K. Echihabi, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art," *Proc. VLDB Endowment*, vol. 12, pp. 112–127, 2018.

[9] K. Echihabi, T. Tsandilas, A. Gogolou, A. Bezerianos, and T. Palpanas, "ProS: Data series progressive k-NN similarity search and classification with probabilistic quality guarantees," *VLDB J.*, 2022, doi: 10.1007/s00778-022-00771-z.

[10] J. Shieh and E. Keogh, "iSAX: Indexing and mining terabyte sized time series," in *Proc. 14th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2008, pp. 623–631.

[11] B. Peng, P. Fatourou, and T. Palpanas, "Fast data series indexing for in-memory data," *Int. J. Very Large Data Bases*, vol. 30, pp. 1041–1067, 2021.

[12] O. Levchenko et al., "BestNeighbor: Efficient evaluation of kNN queries on large time series databases," *Knowl. Inf. Syst.*, vol. 63, pp. 349–378, 2021.

[13] K. Echihabi, K. Zoumpatianos, and T. Palpanas, "Scalable machine learning on high-dimensional vectors: From data series to deep network embeddings," in *Proc. 10th Int. Conf. Web Intell. Mining Semantics*, 2020, pp. 1–6.

[14] J. Wang, Z. Wang, J. Li, and J. Wu, "Multilevel wavelet decomposition network for interpretable time series analysis," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 2437–2446.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 630–645.

[16] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*.

[17] A. Dosovitskiy et al., "An image is worth 16x16 words: Transformers for image recognition at scale," in *Proc. Int. Conf. Learn. Representations*, 2021.

[18] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, Art. no. 418.

[19] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics Intell. Lab. Syst.*, vol. 2, pp. 37–52, 1987.

[20] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas, "Coconut: A scalable bottom-up approach for building data series indexes," *Proc. VLDB Endowment*, vol. 11, pp. 677–690, 2018.

[21] P. Malhotra, V. TV, L. Vig, P. Agarwal, and G. M. Shroff, "TimeNet: Pre-trained deep recurrent neural network for time series classification," in *Proc. Eur. Symp. Artif. Neural Netw.*, 2017.

[22] Q. Wang and T. Palpanas, "Deep learning embeddings for data series similarity search," in *Proc. 27th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 1708–1716.

[23] Y. A. Malkov and D. A. Yashunin, "Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 4, pp. 824–836, Apr. 2020.

[24] A. Babenko and V. S. Lempitsky, "The inverted multi-index," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 6, pp. 1247–1260, Jun. 2015.

[25] Q. Huang, J. Feng, Y. Zhang, Q. Fang, and W. Ng, "Query-aware locality-sensitive hashing for approximate nearest neighbor search," *Proc. VLDB Endowment*, vol. 9, pp. 1–12, 2015.

[26] H. Ferhatosmanoglu, E. Tuncel, D. Agrawal, and A. E. Abbadi, "Vector approximation based indexing for non-uniform high dimensional data sets," in *Proc. 9th Int. Conf. Inf. Knowl. Manage.*, 2000, pp. 202–209.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.

[27] Y. Wang, P. Wang, J. Pei, W. Wang, and S. Huang, "A data-adaptive and dynamic segmentation index for whole matching on time series," *Proc. VLDB Endowment*, vol. 6, pp. 793–804, 2013.
[28] A. Camerra, J. Shieh, T. Palpanas, T. Rakthanmanon, and E. Keogh, "Beyond one billion time series: Indexing and mining very large time series collections with iSAX2," *Knowl. Inf. Syst.*, vol. 39, pp. 123–151, 2014.
[29] K. Zoumpatianos, S. Idreos, and T. Palpanas, "ADS: The adaptive data series index," *Int. J. Very Large Data Bases*, vol. 25, pp. 843–866, 2016.
[30] M. Linardi and T. Palpanas, "Scalable data series subsequence matching with ULISSE," *Int. J. Very Large Data Bases*, vol. 29, pp. 1449–1474, 2020.
[31] D. E. Yagoubi, R. Akbarinia, F. Masseglia, and T. Palpanas, "Massively distributed time series indexing and querying," *IEEE Trans. Knowl. Data Eng.*, vol. 32, no. 1, pp. 108–120, Jan. 2020.
[32] M. Chatzakis, P. Fatourou, E. Kosmas, T. Palpanas, and B. Peng, "Odyssey: A journey in the land of distributed data series similarity search," *Proc. VLDB Endowment*, vol. 16, pp. 1140–1153, 2023.
[33] B. Peng, P. Fatourou, and T. Palpanas, "ParIS: The next destination for fast data series indexing and query answering," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 791–800.
[34] B. Peng, P. Fatourou, and T. Palpanas, "SING: Sequence indexing using GPUs," in *Proc. IEEE 37th Int. Conf. Data Eng.*, 2021, pp. 1883–1888.
[35] Z. Wang, Q. Wang, P. Wang, T. Palpanas, and W. Wang, "Dumpy: A compact and adaptive index for large data series collections," *Proc. ACM Manage. Data*, May 2023, Art. no. 111, doi: 10.1145/3588965.
[36] K. Echihabi, P. Fatourou, K. Zoumpatianos, T. Palpanas, and H. Benbrahim, "Hercules against data series similarity search," *Proc. VLDB Endowment*, vol. 15, pp. 2005–2018, 2022.
[37] I. Azizi, K. Echihabi, and T. Palpanas, "ELPIS: Graph-based similarity search for scalable data science," *Proc. VLDB Endowment*, vol. 16, no. 6, pp. 1548–1559, 2023.
[38] H. Kondylakis, N. Dayan, K. Zoumpatianos, and T. Palpanas, "Coconut: Sortable summarizations for scalable indexes over static and streaming data series," *VLDB J.*, vol. 28, pp. 847–869, 2019.
[39] L. Wu, I. E.-H. Yen, J. Yi, F. Xu, Q. Lei, and M. Witbrock, "Random warping series: A random features method for time-series embedding," 2018, *arXiv:1809.05259*.
[40] Q. Lei, J. Yi, R. Vaculin, L. Wu, and I. S. Dhillon, "Similarity preserving representation learning for time series clustering," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 2845–2851.
[41] J. Paparrizos and M. J. Franklin, "GRAIL: Efficient time-series representation learning," *Proc. VLDB Endowment*, vol. 12, pp. 1762–1777, 2019.
[42] A. Vaswani et al., "Attention is all you need," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 6000–6010.
[43] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, and C. Eickhoff, "A transformer-based framework for multivariate time series representation learning," in *Proc. 27th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2021, pp. 2114–2124.
[44] L. J. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," 2016, *arXiv:1607.06450*.
[45] G. Wilson, J. R. Doppa, and D. J. Cook, "Multi-source deep domain adaptation with weak supervision for time-series sensor data," in *Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2020, pp. 1768–1778.
[46] C. Wu, R. Manmatha, A. J. Smola, and P. Krähenbühl, "Sampling matters in deep embedding learning," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2859–2867.
[47] H. A. Dau et al., "The UCR time series archive," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019.
[48] H. Jégou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 1, pp. 117–128, Jan. 2011.
[49] M. Li, Y. Zhang, Y. Sun, W. Wang, I. W. Tsang, and X. Lin, "I/O efficient approximate nearest neighbour search based on learned functions," in *Proc. IEEE 36th Int. Conf. Data Eng.*, 2020, pp. 289–300.

**Qitong Wang** received the bachelor's and master's degrees in computer science from Fudan University, China. He is currently working toward the PhD degree in computer science with Université Paris Cité, France, under the supervision of Prof. Themis Palpanas. His research interests lie in the intersections between massive data series analysis systems and modern machine learning tools. He has published several papers in international venues including KDD, VLDB, SIGMOD, DASFAA, etc.

**Themis Palpanas** is director of the Data Intelligence Institute of Paris (diiP), senior member of the French University Institute (IUF), and professor of computer science with Université Paris Cité (France). He is the author of 9 US patents, the recipient of 3 Best Paper awards, and the IBM SUR Award. He has served as editor in chief for *Birth Defects Research*, general chair for VLDB 2013, and associate editor for *IEEE Transactions on Knowledge and Data Engineering* and *Data Science and Engineering* journals, as well as PVLDB 2022, 2019 and 2017.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on May 02,2024 at 07:54:30 UTC from IEEE Xplore. Restrictions apply.