# Deep Learning Embeddings for Data Series Similarity Search

Qitong Wang
Université de Paris, LIPADE
qitong.wang@etu.u-paris.fr

Themis Palpanas
Université de Paris, LIPADE &
French University Institute (IUF)
themis@mi.parisdescartes.fr

## Abstract

A key operation for the (increasingly large) 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. Finally, we propose a new sampling strategy, SEASam, that allows SEAnet to effectively train on massive datasets. Comprehensive experiments on 7 diverse synthetic and real datasets verify the advantages of DEA learned using SEAnet, when compared to other state-of-the-art traditional and DEA solutions, in providing high-quality data series summarizations and similarity search results.

## CCS Concepts

• **Information systems** → **Data management systems**; **Temporal data**; • **Computing methodologies** → **Neural networks**.

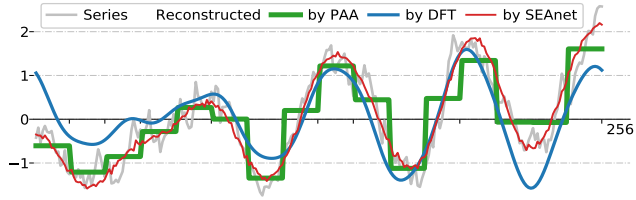
## Keywords

data series; similarity search; indexing; neural networks; sampling

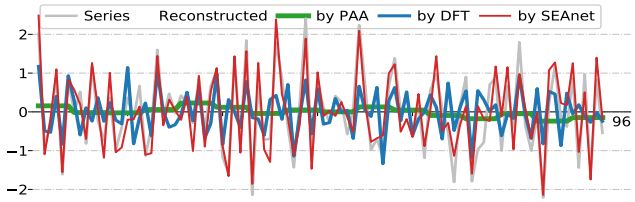
**ACM Reference Format:**
Qitong Wang and Themis Palpanas. 2021. Deep Learning Embeddings for Data Series Similarity Search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '21), August 14–18, 2021, Virtual Event, Singapore.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3447548.3467317

## 1 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 [9, 11, 33, 35]. Typical data series analysis techniques are querying [14], classification [15], clustering [36], anomaly detection [4–6], and visualization [14], 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 [43]. Similarity search can be divided into exact search and approximate search [13]. 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 [14]. Thus, it is very popular in practice, and widely used on massive series collections to enable interactive data exploration and other latency-bounded applications [18]. In this work, we focus on approximate similarity search under Euclidean distance.




**(a) PAA works to approximate and reconstruct a RandWalk series**




**(b) PAA fails to approximate and reconstruct a Deep1B series**

**Figure 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 [10, 12–14]. Most indexes are based on summarized representations of the data series [43] of lower dimensionality[2]. Symbolic Aggregate approXimation (SAX) [28] is a popular and effective discretized summarization. SAX-based indexes [34] are the state-of-the-art (SOTA) data series similarity search methods [13, 14].

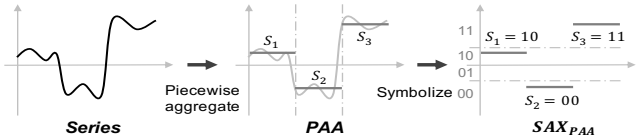
Nevertheless, SAX-based indexes suffer from the problem that SAX fails in hard datasets with specific properties [25]. Since SAX is the symbolization of Piecewise Aggregate Approximation (PAA) [28],

[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 [35].

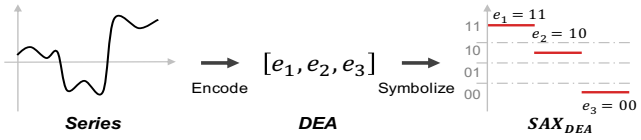
[2] In the data-series literature, *dimensionality* is used interchangeably with *length*, to refer to the number of values of a univariate series.


*KDD '21, August 14–18, 2021, Virtual Event, Singapore*

ACM ISBN 978-1-4503-8332-5/21/08…$15.00
https://doi.org/10.1145/3447548.3467317




(a) PAA-based SAX symbolization




(b) DEA-based SAX symbolization

**Figure 2: Replace PAA by DEA for SAX symbolization.**

failure of PAA to correctly represent some data series directly translates to failure of the PAA-based SAX. Figure 1 illustrates a working and a failing case. The high frequency of the Deep1B series (Figure 1b) implies more cycles than the available SAX words: each PAA segment has to average values over ≥1 cycles, leading to similar PAA values across different segments, and to indistinguishable SAX words across different series. Introducing more SAX words 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 [3], is to learn vectors possessing necessary latent information for classification, clustering and other downstream applications. Its success in data series has also been reported for speech recognition [27], data series classification [17] and many other applications. Embedding techniques have been proven to be capable of capturing frequency [41] 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 Figure 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 [19]. It adopts the idea of exponentially increasing dilations, which has been verified to be efficient for data series applications [2]. In contrast to existing convolutional autoencoders for series embedding [17], 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) [45]. 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 SEA-Sampling (SEASam), a novel sampling strategy based on a sortable data series summarization [23]. SEAsam enables SEAnet (and other networks) to effectively fit a large dataset, leading to improved performance.

Comprehensive experiments verify that, compared to PAA and DEA generated by other SOTA architectures, including FDJNet [17], TimeNet [31] and InceptionTime [15], the DEA generated by SEAnet is more effective in preserving the original pairwise distances in the lower-dimensional summarized space. This advantage also translates to more accurate approximate similarity search across several synthetic and real data series collections with diverse properties.

**[Contributions]** Our contributions are summarized as follows.

(1) We propose the use of deep learning embedding techniques to 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.

(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) A novel sampling strategy, SEAsam, is proposed to draw representative samples from massive data series collections, enabling effective training for the deep models.

(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.

(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 [42].

## 2 Background and Related Work

A **data series**, $S = \{p_1, ..., p_m\}$, 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 data series. $\mathcal{S}$ denotes a collection of data series, i.e., $\mathcal{S} = \{S_1, ..., S_n\}$. We call $n$ the *size* of the data series collection. A **summarization** $E = \{e_1, ..., e_l\}$ 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** $d$ we use is Euclidean distance, which is a widely adopted and effective measure for data series similarity search [43].$d'$ 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 closest series $S_c$, **approximate similarity search** targets to find a series $S'_c \in \mathcal{S}$ such that $d(S'_c, S_q) \approx d(S_c, S_q)$. $\frac{d(S_c,S_q)}{d(S'_c,S_q)} \in (0, 1]$ is called $S'_c$'s **tightness**.

The most prominent **data series indexing** techniques can be categorized into optimized scans [16], and tree-based indexes [40]. Recent studies [13, 14] have demonstrated that the SAX-based indexes [34] achieve SOTA performance under several conditions. In this work, we use MESSI as our iSAX index [37, 38], because its main-memory operation and parallel design lead to SOTA performance.

**[Learned Data Series Embeddings]** Deep representation learning has been popular and successful in several domains [8, 22, 27]. On the other hand, few recent works [17, 31] focus on data series representation learning, none of which targets similarity search.

Autoencoder is a category of deep neural networks to learn embeddings [3]. 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 [41].

TimeNet [31] and FDJNet [17] are two SOTA architectures for data series representation learning. TimeNet deploys a multi-layer GRU straightforwardly to embed and reconstruct series. FDJNet is based on Temporal Convolutional Network (TCN) [2] to embed series. Its core design choices are (1) to increase dilations exponentially in deeper layers; (2) to remove dependence on values from future positions during convolution. However, neither TimeNet nor FDJNet has been evaluated for similarity search before.

Aside from representation learning, deep models are exploited in other data series applications. The SOTA series classification method, InceptionTime [15] adapts Inception for series classification. However, information learned from different scales in the Inception module might perturb the distance space structure.

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.

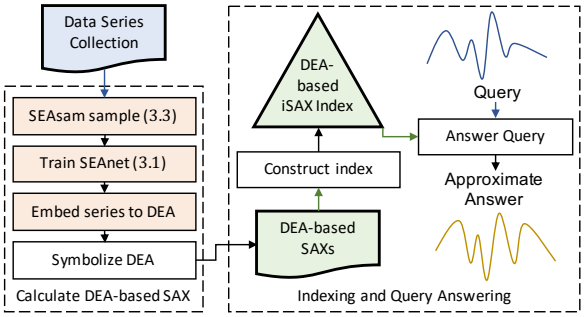



**Figure 3: Workflow of DEA-based approximate similarity search.**

## 3 DEA-based Approximate 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 Figure 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 structured into an iSAX index, where approximate similarity search can be efficiently conducted.

SEAnet is a novel autoencoder proposed to learn high-quality DEA. Unlike FDJNet [17] and other SOTA convolutional architectures for series embedding, SEAnet is composed of both an 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). The necessity of decoders has also been observed by other works [44]. SEAnet stacks dilated full-preactivation ResBlocks [19]. 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 3.1, and further discuss the SoS preservation principle in Section 3.2.

Our SEAsam strategy makes use of the inverse iSAX sortable summarization [23]. 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 [23]. Thus, sampling proceeds by drawing series of equal intervals from dataset sorted in InvSAX order (cf. Section 3.3).

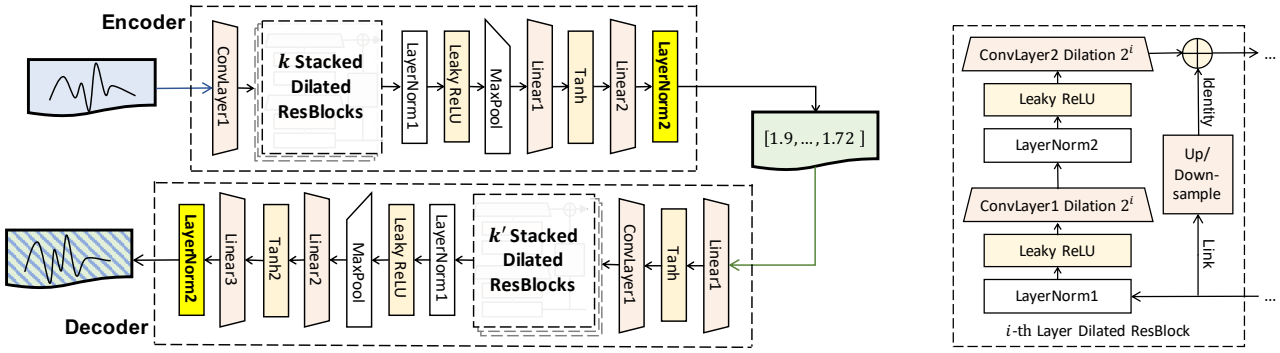
Since DEA acts as a replacement for PAA, the high-level indexing and query answering procedures remain similar[3] to an ordinary iSAX. Considering the low-latency requirement of approximate similarity search applications, our design follows MESSI [37, 38], the SOTA concurrent in-memory iSAX index.

### 3.1 SEAnet Architecture

The SEAnet architecture is illustrated in Figure 4a. It comprises a convolutional encoder and a homogeneous decoder. We first give an overview of the architecture, and then present its details.

The first part of the SEAnet encoder, from ConvLayer1 to MaxPool, comprises $k$ stacked dilated full-preactivation ResBlocks [19]

[3]The only difference is that DEA distances cannot lower bound the target distance measures. Addressing this issue is part of our future work.




(a) SEAnet Architecture (b) Dilated ResBlock in SEAnet

**Figure 4: The SEAnet architecture and the details of a dilated full-preactivation ResBlock.**

for nonlinear transformations. The dilated ResBlock is illustrated separately in Figure 4b. 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 [2]. 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 [17], the SEAnet encoder is finalized by LayerNorm2, which is specifically designed using the SoS preservation principle. We elaborate on this in Section 3.2.

The SEAnet decoder corresponds to the encoder, except for a preceding Tanh-activated linear layer, introduced to adjust dimensionality. Similar to previous studies [30], we claim that the encoder and decoder need not be homogeneous. Although encoder-only architectures is the popular choice [17], 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.

**[Training Procedure]** We first provide the intuitions behind the SEAnet training, and then the mathematical formalizations.

SEAnet is trained in a pairwise manner 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 rational behind these steps is further explained in Section 3.2.

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 (being treated as constants instead of input 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 $L_R$.

## 3.2 Sum of Squares Preservation

We propose a SoS preservation framework for effective DEA learning. SoS preservation has been observed before [45], but to the best of our knowledge, has never been formally introduced to representation learning. Given an $n \times m$ matrix $M$, where each row $M_{i,*}$ corresponds to a series and each column $M_{*,j}$ corresponds to a position, SoS $= \sum_{i,j} M_{i,j}^2$. Note that defining new axes based on the largest SoS is equivalent to selecting the largest eigenvalues in linear dimensionality reductions on z-normalized datasets, with the purpose of preserving information about the dataset through linear transformations [45]. Thus, SoS could be regarded as an indicator of the transformation quality. By keeping SoS invariant, the quality of DEAs are upheld from this perspective, and the networks then 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 [1]. We now elaborate on the design of SEAnet under this principle.

Considering the fact that z-normalizing data series is a very common operation [7], we constrain SEAnet to keep SoS invariant

| | Before Scaling | | Scaled by $\sqrt{256/m}$ | | Scaled by $\sqrt{1/m}$ | |
|---|---|---|---|---|---|---|
| Length $m$ | 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 |

**Table 1: Mean and Variance of the distribution of pairwise distances between two ideal series. 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 DEA in loss functions. (Note that for length 256, scaling by $\sqrt{256/m}$ does not change the original behavior.)**

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 $\frac{\sqrt{m}}{\sqrt{l}}$.

**[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 [1]. Latent variables with $\mu = 0$ and $\sigma = 1$ are widely considered among the best choices for the gradients' back-propagation [1, 19]. Hence, we keep the DEAs z-normalized, and scale the series by $\frac{1}{\sqrt{m}}$ and DEA by $\frac{1}{\sqrt{l}}$ in $L_C$ and $L_R$.

Typical examples of scaling are presented in Table 1. We make three observations: (1) After scaling short series by $\sqrt{\frac{256}{m}}$, 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).

### 3.3 Sampling with SEAsam

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 we need our sample to 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 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; this solution though, would be prohibitively expensive.

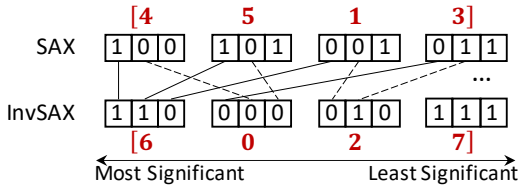



**Figure 5: SEAsam transformation and InvSAX [23].**

To this end, we propose SEAsam (SEA Sampling), a novel data series sampling strategy based on the sortable data series representation, InvSAX [23]. 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) [40]. 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 Figure 5. 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 $[6, 2, 2, 6]$.

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 $O(nm)$, and the space complexity of SEAsam is $O(nl)$, rendering SEAsam an efficient strategy.

## 4 Experimental Evaluation

We present our experimental evaluation of SEAnet, DEA-based data series similarity search, and SEAsam 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 @ 3.20GHz and 320GB 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 [37] 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 [25]. They were generated through Inverse Discrete Fourier Transform (IDFT) from a random spectrum with its first 5 or 10 components being amplified. Four real datasets are Seismic from seismology, Astro from astronomy, SALD from neuroscience and Deep1B from image processing [14]. Length of each series is 128 for SALD, 96 for Deep1B, and 256 for the rest. We note that these datasets are considered hard for the similarity search task [13, 14]. We experimented with datasets ranging between 1M to 100M series[4].

**[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 [17], TimeNet [31], and InceptionTime [15]. We describe these methods below. SEAsam was compared to uniformly random sampling.

PAA [21] is the baseline method to evaluate DEA's summarization quality. PAA-based MESSI iSAX index [37, 38] 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 4. We used the same convention for other methods, as well, e.g., SEAnet 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 [31]. For InceptionTime, we used the same structure as SEAnet, but replaced ResBlock with InceptionBlock [15]. 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 kept the same to series length following the conventions [2].

All models were trained using SGD and the same loss function (cf. Section 3.2). Training size was 200,000 series, and validation size was 20,000. TimeNet was trained for 125 epochs, while the others for 100 epochs. SEAnet, SEAnet-nD, FDJNet and InceptionTime were initialized by LSUV [32], while TimeNet was initialized by default. 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 {1$e$-3, 5$e$-3, 1$e$-2, 2.5$e$-2, 5$e$-2}, and was either linearly decayed (every epoch), or exponentially decayed (by 0.9 every 2 epochs) until 1$e$-5. Totally, 5,040 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.

| | *Length* | SEAnet | SEAnet-nD | FDJNet | TimeNet | Incept |
|---|---|---|---|---|---|---|
| parameters (millions) | *96* | 0.585 | 0.288 | 0.262 | 0.746 | 0.512 |
| | *128* | 1.234 | 0.609 | 0.563 | 1.32 | 0.832 |
| | *256* | 5.712 | 2.823 | 2.634 | 5.23 | 2.126 |
| MACs (billions) | *96* | 0.053 | 0.027 | 0.028 | 0.07 | 0.085 |
| | *128* | 0.152 | 0.076 | 0.082 | 0.17 | 0.15 |
| | *256* | 1.412 | 0.706 | 0.756 | 1.34 | 0.49 |
| Seconds / Epoch | *96* | 87.8 | 46.7 | 19.8 | 100.8 | 223.1 |
| | *128* | 93.7 | 52.4 | 23.5 | 180.1 | 365.6 |
| | *256* | 312.1 | 173.4 | 130.7 | 1044.8 | 905.2 |

**Table 2: The complexities of the deep models, in terms of number of tunable parameters (in millions), Multiply-and-Accumulate (MAC) (in billions), and training time per epoch (in seconds).**

Architecture complexities are summarized in Table 2, 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 Table 2.

**[Measures]** To evaluate summarization quality, we used the following 3 measures, i.e., average distance differences (unscaled $L_C$), reconstruction RMS, and NN coverages. Series subsets or series pairs were SEAsam samples from 100M datasets. PAA is implied as a special case of DEA if without ambiguity.

(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 Figure 6.)

(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 $S' = [..., p'_i, ...]$ is the reconstruction of series $S = [..., p_i, ...]$. Reported values were averaged from 20,000 series.

(3) **NN Coverages**. 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 the direct measure of whether the structure of original distance spaces is preserved or not. We reported NN coverages for $k \in \{1, 5, 10, 50, 100, 500, 1{,}000\}$ in Section 4.2. 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) [14]. 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 (see Sections 4.1 and 4.3), we report the 1st BSF tightness as a function of the number of series examined (this makes for a fair comparison across indices, which may have leaves containing different number of series). Similar to previous work, we report the average tightness over 1,000 queries [13, 14].

[4] 100M series of length 256 $\approx 100GB$.

## 4.1 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. Detailed results are omitted for brevity.

**[SoS Preservation]** First, we evaluate the effect of the scaling steps we introduced under the SoS preservation. 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 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 3.2), or did not converge and generate random DEAs (exhibiting similar statistics to Table 1).

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. 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%), SESASam 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. 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 synthetic dataset F10.

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

## 4.2 DEA Quality

**[Average Distance Differences]** The averaged distance differences reported in Table 3a, 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 best averaged difference on the Astro dataset 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 Figure 6. 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 3b. 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 Figure 7. 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 was not as obvious as in Table 3a, except for SEAnet, indicating that the target of preserving original distances in DEA distances alone cannot guarantee to provide high-quality DEAs for similarity search. This observation, together with the fact that SEAnet outperformed SEAnet-nD, verifies 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 Figure 7e 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.

| | (a) Averaged Distance Differences | | | | | | (b) Reconstruction Root-Mean-Square Error | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Dataset | PAA | FDJNet | TimeNet | InceptionTime | SEAnet-nD | SEAnet | PAA | TimeNet | InceptionTime | 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 |

**Table 3: (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. (Results calculated using 10,000 series SEAsam sampled from datasets of 10-million series.)**

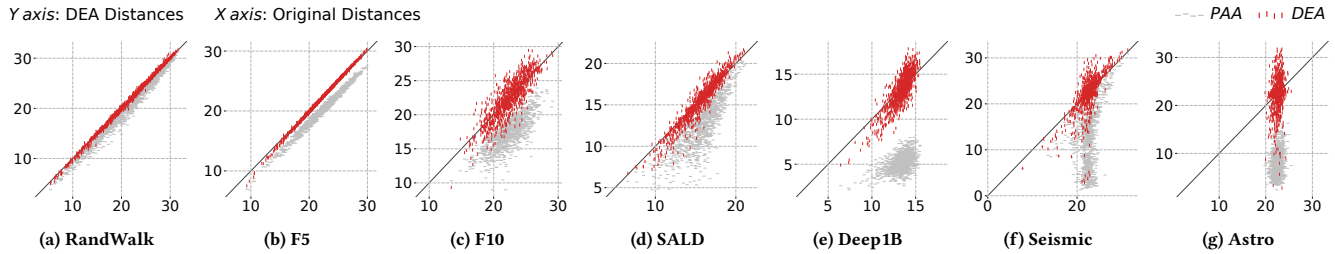



**Figure 6: Distance scatters of 1,000 SEAsam sampled series pairs across different datasets for PAA (light gray) and SEAnet (dark red).**

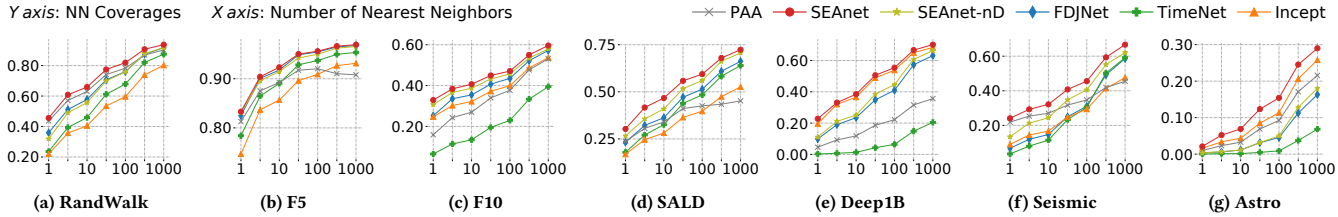



**Figure 7: Nearest neighbors' coverage vs neighborhood size (higher is better).**

## 4.3 DEA for Approximate 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 Figure 8. 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.

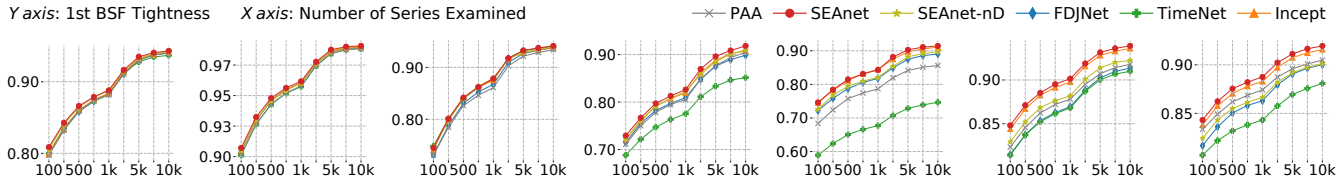
Besides the 1st BSF tightness, the index's leaf node compactness (i.e., the average distances among the series in a leaf node) also profiles the quality of the index built on DEA or PAA. Smaller average distance indicates the index is more successful in grouping similar series into the same leaf node.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

**[1st BSF Tightness across DEA lengths]** We also measured the 1st BSF tightness with varying DEA and PAA representation lengths between 8-16 values. SEAnet performed better than the competitors across all representation lengths and all datasets (we omit the detailed results due to lack of space).

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

## 5 Discussion and Conclusions

In this paper, we introduce the use of deep learning embeddings, DEA, for data series similarity search. We propose a novel autoencoder, SEAnet, designed under the firstly introduced SoS preservation principle, for effectively learning DEA. A new sampling strategy, SEAsam, is introduced in order to facilitate SEAnet's training on massive collections. We demonstrate that the DEA learned by SEAnet 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 (when examining either a small, or a large number of candidates). 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




**(a) RandWalk** **(b) F5** **(c) F10** **(d) SALD** **(e) Deep1B** **(f) Seismic** **(g) Astro**

**Figure 8: Approximate query answers quality: 1st BSF tightness vs number of series visited; 100M series datasets (higher is better)**

dynamic datasets, the development of more powerful sampling strategies, and the careful study of query answering strategies on top of DEA, including *product quantization* [20], *locality sensitive hashing* [26], and modern data series indexes [24, 29, 39].

**[Acknowledgements]** Work supported by Investir l'Avenir and Université de Paris IDEX Emergence en Recherche ANR-18-IDEX-000, Chinese Scholarship Council, FMJH Program PGMO, EDF, Thales, HIPEAC 4, GENCI–IDRIS (Grant 2020-101471), and NVIDIA Corporation for the Titan Xp GPU donation used in this research.

## References

[1] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer Normalization. *arXiv* (2016).
[2] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. 2018. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling. *arXiv* (2018).
[3] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation Learning: A Review and New Perspectives. *PAMI* (2013).
[4] Paul Boniol, Michele Linardi, Federico Roncallo, Themis Palpanas, Mohammed Meftah, and Emmanuel Remy. 2021. Unsupervised and Scalable Subsequence Anomaly Detectionin Large Data Series. *VLDBJ* (2021).
[5] Paul Boniol and Themis Palpanas. 2020. Series2Graph: Graph-based Subsequence Anomaly Detection for Time Series. *PVLDB* 13, 11 (2020).
[6] Paul Boniol, John Paparrizos, Themis Palpanas, and Michael J. Franklin. 2021. SAND: Streaming Subsequence Anomaly Detection. *PVLDB* (2021).
[7] Hoang Anh Dau, Anthony J. Bagnall, Kaveh Kamgar, Chin-Chia Michael Yeh, Yan Zhu, Shaghayegh Gharghabi, Chotirat Ann Ratanamahatana, and Eamonn J. Keogh. 2019. The UCR time series archive. *JAS* (2019).
[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*.
[9] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2020. Big Sequence Management: on Scalability. In *IEEE BigData*.
[10] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. Big Sequence Management: Scaling up and Out. In *EDBT*.
[11] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. High-Dimensional Similarity Search for Scalable Data Science. In *ICDE*.
[12] Karima Echihabi, Kostas Zoumpatianos, and Themis Palpanas. 2021. New Trends in High-D Vector Similarity Search: AI-driven, Progressive, and Distributed. In *VLDB*.
[13] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2018. The lernaean hydra of data series similarity search: An experimental evaluation of the state of the art. *VLDB* (2018).
[14] Karima Echihabi, Kostas Zoumpatianos, Themis Palpanas, and Houda Benbrahim. 2019. Return of the Lernaean Hydra: experimental evaluation of data series approximate similarity search. *VLDB* (2019).
[15] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for time series classification. *DMKD* (2020).
[16] Hakan Ferhatosmanoglu, Ertem Tuncel, Divyakant Agrawal, and Amr El Abbadi. 2000. Vector Approximation based Indexing for Non-uniform High Dimensional Data Sets. In *CIKM*.
[17] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. 2019. Unsupervised scalable representation learning for multivariate time series. In *NeurIPS*.
[18] Anna Gogolou, Theophanis Tsandilas, Karima Echihabi, Anastasia Bezerianos, and Themis Palpanas. 2020. Data Series Progressive Similarity Search with Probabilistic Quality Guarantees. In *SIGMOD*.
[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity Mappings in Deep Residual Networks. In *ECCV*.
[20] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *PAMI* (2011).
[21] Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. 2001. Locally adaptive dimensionality reduction for indexing large time series databases. In *SIGMOD*.
[22] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. 2019. Revisiting Self-Supervised Visual Representation Learning. In *CVPR*.
[23] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. 2018. Coconut: A Scalable Bottom-Up Approach for Building Data Series Indexes. *VLDB* (2018).
[24] Haridimos Kondylakis, Niv Dayan, Kostas Zoumpatianos, and Themis Palpanas. 2019. Coconut: Sortable summarizations for scalable indexes over static and streaming data series. *VLDBJ* 28, 6 (2019).
[25] Oleksandra Levchenko, Boyan Kolev, Djamel Edine Yagoubi, Reza Akbarinia, Florent Masseglia, Themis Palpanas, Dennis Shasha, and Patrick Valduriez. 2020. BestNeighbor: Efficient Evaluation of kNN Queries on Large Time Series Databases. *KAIS* (2020).
[26] Mingjie Li, Ying Zhang, Yifang Sun, Wei Wang, Ivor W. Tsang, and Xuemin Lin. 2020. I/O Efficient Approximate Nearest Neighbour Search based on Learned Functions. In *ICDE*.
[27] Runnan Li, Zhiyong Wu, Jia Jia, Yaohua Bu, Sheng Zhao, and Helen Meng. 2019. Towards Discriminative Representation Learning for Speech Emotion Recognition. In *IJCAI*.
[28] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *SIGMOD*.
[29] Michele Linardi and Themis Palpanas. 2020. Scalable Data Series Subsequence Matching with ULISSE. *VLDBJ* (2020).
[30] Yukun Ma, Jiu Xu, Björn Stenger, Chen Liu, and Yu Hirate. 2018. Deep Heterogeneous Autoencoders for Collaborative Filtering. In *ICDM*.
[31] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam M. Shroff. 2017. TimeNet: Pre-trained deep recurrent neural network for time series classification. In *ESANN*.
[32] Dmytro Mishkin and Jiri Matas. 2016. All you need is a good init. In *ICLR*.
[33] Themis Palpanas. 2015. Data series management: The road to big sequence analytics. *SIGMOD Record* (2015).
[34] Themis Palpanas. 2019. Evolution of a Data Series Index. In *ISIP*.
[35] Themis Palpanas and Volker Beckmann. 2019. Report on the first and second interdisciplinary time series analysis workshop (itisa). *SIGMOD Record* (2019).
[36] John Paparrizos and Luis Gravano. 2015. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*.
[37] Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2020. MESSI: In-Memory Data Series Indexing. ICDE.
[38] Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2021. Fast Data Series Indexing for In-Memory Data. *VLDBJ* (2021).
[39] Botao Peng, Panagiota Fatourou, and Themis Palpanas. 2021. SING: Sequence Indexing Using GPUs. In *ICDE*.
[40] Jin Shieh and Eamonn Keogh. 2008. iSAX: indexing and mining terabyte sized time series. In *KDD*.
[41] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel Wavelet Decomposition Network for Interpretable Time Series Analysis. In *KDD*.
[42] Qitong Wang and Themis Palpanas. 2021. *SEAnet homepage.* https://helios.mi.parisdescartes.fr/~themisp/seanet/
[43] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn J. Keogh. 2013. Experimental comparison of representation methods and distance measures for time series data. *DMKD* (2013).
[44] Zbigniew Wojna, Vittorio Ferrari, Sergio Guadarrama, Nathan Silberman, Liang-Chieh Chen, Alireza Fathi, and Jasper Uijlings. 2019. The devil is in the decoder: Classification, regression and gans. *IJCV* (2019).
[45] Svante Wold, Kim Esbensen, and Paul Geladi. 1987. Principal component analysis. *Chemometrics and intelligent laboratory systems* (1987).