On Convolutional Autoencoders to Speedup Similarity-Based Time Series Mining

Yuri Gabriel Aragão da Silva
Departamento de Computação
Universidade Federal de São Carlos, Brazil
Email: yuri.aragao@estudante.ufscar.br

Diego Furtado Silva
Departamento de Computação
Universidade Federal de São Carlos, Brazil
Email: diegofs@ufscar.br

***Abstract*—Time series represent a type of data that is increasingly present in research and industry applications. The most commonly used approach to obtain knowledge from these data is similarity-based data mining algorithms. However, in large volumes of data, applying these algorithms may be infeasible. Therefore, there are several techniques for accelerating the distance calculation between time series proposed in the literature, such as algorithms' adaptations, indexing, and approximations. Recent results show that the union of techniques in different categories usually improves efficiency in time series mining tasks by similarity. In this way, we propose a method to speed up time series distance calculation based on dimensionality reduction through convolutional autoencoders. To avoid the neural network training performs as a bottleneck in the data mining's runtime, we propose an offline training phase, with previously known data. Our proposal is orthogonal to state-of-the-art tools so we can apply it as a complementary technique. We show that our proposal can lead similarity-based algorithms to execute up to two orders of magnitude faster than these tools alone, without loss of quality in the results obtained through three different tasks.**

I. Introduction

The lower prices and wide distribution that sensors and mobile devices achieved in recent decades have made terms such as the internet of things and industry 4.0 well-known [1]. Similarly, the variety and high availability of storage devices have allowed us to store larger volumes of data to monitor diverse activities [2]. The myriad of applications in these scenarios have one thing in common: increasing volumes of time series data to represent the analyzed phenomena.

These data allow the use of data mining algorithms to obtain knowledge regarding its variation over time. Data mining can be applied in the time series domain in general tasks, such as clustering and classification, but also domain-specific tasks, such as motif discovery, anomaly detection, and semantic segmentation.

One of the most common approaches to perform these tasks is through a similarity-based algorithm [3]. Several tasks, such as motifs discovery, have the concept of similarity or distance, in their definitions. Besides, similarity-based algorithms achieved excellent results in other applications, such as classification and anomaly detection.

A recurring concern in this context is the computational cost. If we rely on straightforward implementations, calculating distances can represent a bottleneck in the time spent to execute these algorithms, making it impossible to scale them for large volumes of data [4].

Therefore, several speedup techniques of similarity-based algorithms have been proposed for time series [5], [6]. These proposals can apply indexing techniques, approximations, dimensionality reduction, or modifications in the mining algorithm. In some cases, the algorithms that stand out for their efficiency use more than one of these approaches.

We explore the application of deep convolutional autoencoders (CAE) to reduce the dimensionality of time series. We assess their suitability for speeding up similarity-based mining algorithms through experiments in three different tasks and compared the execution time with and without the use of autoencoders.

As training the autoencoder may compromise the runtime of these tasks, we propose a two-steps procedure, illustrated in Figure 1. In the offline step, we consider previously know data to train the CAE. Then, we evaluate its application in a set of time series that did not belong to the training dataset.

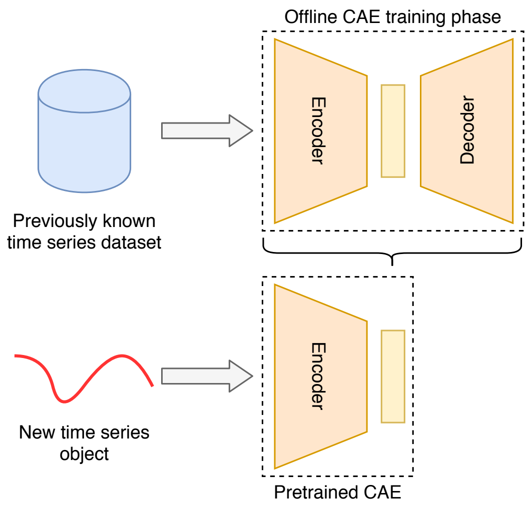

Offline CAE training phase
Encoder
Decoder
Previously known time series dataset
New time series object
Encoder
Pretrained CAE

Fig. 1: Our proposal consists of training a convolutional autoencoder in an offline phase and, later, applying it in a new task to speed up the similarity calculation.

In our empirical evaluation, we used CAE in conjunction with other speedup techniques. Our results shows that these networks can achieve good generalization with a proper offline training, allowing speedup of up to 400 times in the evaluated tasks, without significant losses of similarity relationships between the series.

The remaining of this paper is organized as follow. Section II briefly presents a necessary background and research efforts on large-scale similarity-based time series mining algorithms. Section III presents the procedure proposed to train and use convolutional autoencoders to accomplish our goals. Section IV defines the experimental setup we used to evaluate our proposal. The results obtained in our experiments are presented and discussed in Section V. Finally, Section VI presents the final remarks of this work.

II. Background and Related Work

A time series x of length n is a set of ordered values such that $x = (x_1, x_2, \ldots, x_n)$, $x_i \in \mathbb{R}, \forall i \in 1, 2, \ldots, n$. We refer to each value x_i as an observation of the time series. This definition applies to other sequential data not collected over time, like the contour of figures described as a vector. Besides, we assume that the consecutive values of x are equally spaced, i.e., the time (or space) difference between successive observations is not relevant.

There is a multitude of distance measurements that have been used successfully in time series mining tasks [7], [8], [9]. The most commonly used ones are Euclidean Distance (ED) and Dynamic Time Warping (DTW). The main difference between these measures is the way they view the alignment between the series' observations. The ED, defined by Equation 1, considers a linear alignment between the series.

$$ed(x, y) = \sqrt{\sum_{k=1}^{n} (x_k - y_k)^2} \tag{1}$$

where x nd y are equal-length time series.

On the other hand, the DTW applies a dynamic programming algorithm to obtain the best alignment between the observations, considering boundaries, continuity, and monotonicity constraints [10]. Equation 2 defines the recurrence relation used to calculate DTW.

$$dtw(i, j) = c(x_i, y_j) + min \begin{cases} dtw(i-1, j) \\ dtw(i, j-1) \\ dtw(i-1, j-1) \end{cases} \tag{2}$$

where $c(x_i, y_j) = (x_i - y_j)^2$. The distance between these time series is given by $dtw(n, m)$, where n and m are the lengths of x and y, respectively. Limiting the difference between the i and j values, technique know as warping window [11], is a good practice when applying DTW.

The straightforward calculation of the distances can compromise the adopted solution's viability in large-scale applications. Several algorithms seek to reduce the runtime through different strategies to overcome this limitation. We can categorize these proposals into three main groups: indexing [12], task-dependent or independent algorithmic improvement [13], [5], and dimensionality reduction [10], [14], [15]. Besides, there are approximation algorithms [16], but these techniques can significantly distort the distances between the analyzed series and not provide improvements in runtime [17].

Since the ED is a metric, we can use any indexing technique in the metric space o speed up the similarity search under this measure. On the other hand, DTW does not respect triangular inequality. Therefore, we need different indexing approaches to reduce its cost. The UCR Suite [18] uses a series of lower bound techniques to index the search for similarity using DTW, making it possible to perform this task on a large scale.

When we refer to strategies that seek improvements in the algorithm used to calculate the distance, we refer to developments independent or not of the task performed. For example, algorithms such as PrunedDTW [19] and SparseDTW [20] exploit features such as the upper bound of the distance and sparsity of the time series to speed up the DTW calculation between a pair of objects. We can use these algorithms in any DTW-based mining task.

The Matrix Profile (MP) is undoubtedly the state-of-the-art approach for many ED-based tasks. Through the nearest neighbor join operation, we can apply this primitive in varied time series mining tasks with little effort [21], such as discovering discords and pair motifs. The primary mechanism behind the algorithms used to obtain the MP is the reuse of scalar products obtained by the convolution between the series [22] and pre-calculating the other statistics necessary for the joining operation. Modifying the straightforward distance calculation between all the subsequences to leverage faster algorithms allows the MP to deal with large volumes of data.

The category of algorithms for speedup used in this work is the dimensionality reduction. Arguably, the most straightforward and widely used algorithms in this category are the Piecewise Aggregation Approximation (PAA) [23] and its variations. This algorithm consists of breaking the time series in uniform intervals and represent each interval by a single value: the mean of the observations within that window. At the end of this procedure, PAA reduces the time series proportionally to the interval length.

Finally, we note that these strategies are orthogonal. That is, we can use techniques in different of these categories in the same task. For example, the USP-UCR Suite tool [5] uses extends the UCR Suite by applying DTW indexing techniques in conjunction with PrunedDTW to speed up the search for similarity by up to five times compared to using only indexing methods. Another example of a hybrid approach is leveraging indexing techniques to provide new perspectives to the MP and make its calculation even more efficient [24].

III. Proposed Method

We propose and evaluate a convolutional autoencoder-based dimensionality reduction to speed up the distance calculation between time series objects. Although the CAEs comprise a well-known set of dimensionality reduction techniques in some domain [25], we are not aware of their use for the task we investigate. CAEs were used as an auxiliary technique in the time series domain to other tasks, such as visualization [26] and outlier detection [27].

One possible reason for this gap is because the CAE training phase can be costly, even with appropriate computational resources. Thus, training and applying the autoencoder when calculating distances can generate an even more significant bottleneck in the execution time. Therefore, we propose training the CAE in an offline phase.

To achieve this goal, we must consider the evaluation phase comprises only new data, i.e., time series that does not belong to the training set. Thus, we can assess the cost of reducing the dimensionality of the series using the CAE and estimating the distance to other objects in this reduced representation. Therefore, in addition to evaluating the improvement in runtime, we need to assess the model's generalizability.

Figure 2 illustrates the architecture we used in our experimental evaluation. It is quite simple and consists of three blocks in the encoding phase, composed of convolutional layers followed by dimensionality reduction. Consequently, we have three pairs of convolution and upsampling layers ending in a final convolution layer in the decoding phase. For scope limitation, we restrict our experiments to the use of one-dimensional time series. Therefore, all convolutional layers are equally one-dimensional. Further details of the hyperparameters used in our evaluation are presented in Section IV-A.

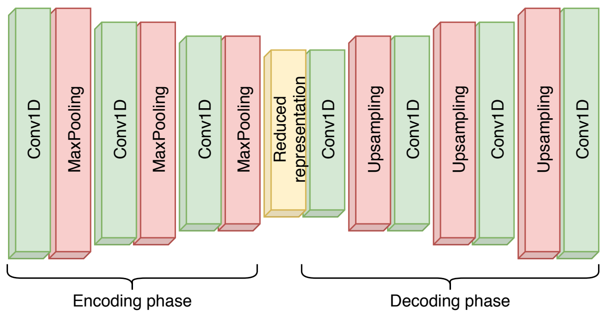


Fig. 2: The CAE used in our experimental evaluation is composed of three blocks of convolutional e downsampling layers in the encoder, and a convolutional layer plus three blocks of upsampling and convolutional layers in the decoder.

After the offline training phase, only the encoding phase continues necessary. With each new time series (or dataset) at hand, we reduce it using the encoder, that is, the layers that make up the CAE encoding phase, before applying any distance function.

IV. Experimental Setup

In this section, we describe the procedures we adopted to assess the proposed method. We consider three different similarity-based tasks and compare the results obtained with the raw time series and the features extracted by the CAE. These tasks are the all-pairwise distance matrix, the similarity search, and the motif and discord discovery (the matrix profile calculation).

Besides, we performed a runtime experiment[1] to measure the gain obtained by extracting features. In this case, we do not consider the training phase as part of the total time, given we perform this step offline. To ensure the correctness of this approach, we train the CAE in separate time series. So, we only take into account the time to extract the features in the pre-trained neural network. The benefits of this procedure are twofold: it ensures the offline training and assesses CAE's generalization ability.

The next sections describe the CAE architecture and the approached tasks in detail, present the used datasets, and describe the procedures to train and evaluate the CAE in each scenario.

A. CAE Configuration

As presented in the Section III, the CAE used in our evaluation has three encoding blocks and four to decode. The convolutional layers in these blocks use kernels of size (1×3), with Swish activation function [28]. The number of applied kernels in each layer is $(16, 8, 1, 8, 8, 16, 1)$, respectively.

Each maxpooling layer reduces the dimension by half. Therefore, the intermediate layer of the network is eight times smaller than the input layer. In the decoding phase, each upsampling layer doubles the previous layer's dimension, so the network output has the same size as the input.

To implement this architecture, we used the Tensorflow 2.0 library [29]. We used The MSE (Mean Squared Error) loss function and Adam optimization algorithm in the training process, with a batch size of 16 and 50 epochs in total. Finally, we use early stopping based on loss values, i.e., if the loss does not decrease in three consecutive epochs, the training ends.

B. All-Pairwise Distance

The distance calculation between all pairs of time series can be used as input for relational clustering algorithms [8], as well as some classification algorithms [30]. Besides, they can be an excellent tool for evaluating speedup techniques [19]. We used this approach in the first – and with the smallest scale – scenario for evaluating our proposal. For this, we limit ourselves to calculating the Euclidean distance.

Figure 3 illustrates the procedure adopted in this task. To evaluate our proposal's suitability for a dataset, we trained the CAE with other sets composed of time series with the same length. After this offline training step, we transform all the series in the new dataset using the CAE and then calculate the all-pairwise distance matrix.

We use datasets contained in the UCR Time Series Classification Archive [31]. As we are not interested in the classification, we disregard the default data separation in training and test sets. Instead, we concatenate all the data to create a larger version of the dataset. We selected groups of datasets in which the time series have the same length. Then, we train the

[1]All experiments were carried out in the same computational environment: a 12-core Intel® Core^{TM} i7, 32 GiB system memory, and one NVIDIA GeForce RTX 2050. Besides, we do not execute any other process parallel to each experiment, except for the processes necessary for the operating system.

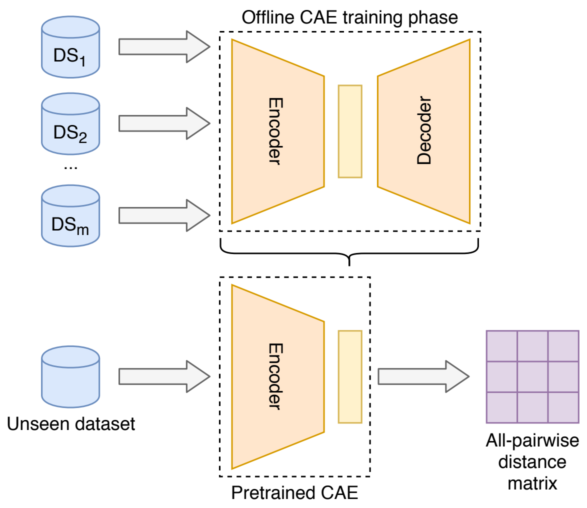


Fig. 3: To calculate the all-pairwise distance, we first train the CAE with different same length time series datasets. Then, we apply CAE to the new dataset and estimate the distance matrix.

CAE in all but one of these datasets. Also, we avoid datasets in which the same instance may appear in both subgroups. For example, the datasets Worms and WormsTwoClasses have the same data, only differing on the label information. So, we do not use these datasets. Table I shows the groups of datasets used in this task.

TABLE I: Groups of datasets for CAE's training and evaluation in the all-pairwise ED calculation task.

ID	Training	Evaluation
AP1	PigArtPressure + PigAirwayPressure	PigCVP
AP2	EOGVerticalSignal	EOGHorizontalSignal
AP3	Mallat + MixedShapes + MixedShapesSmallTrain + Phoneme	StarlightCurves
AP4	NonInvasiveFetalECGThorax1	NonInvasiveFetalECGThorax2
AP5	SmallKitchenAppliances + Computers + RefrigerationDevices + LargeKitchenAppliances	ScreenType
AP6	ShapesAll + Herring	Earthquakes
AP7	Herring	ShapesAll
AP8	Herring + Earthquakes	ShapesAll
AP9	FordA	FordB
AP10	UWaveGestureLibraryX + UWaveGestureLibraryY	UWaveGestureLibraryZ

This setup leads to some interesting evaluation scenarios. For example, we have cases where all the datasets belong to the same phenomenon but monitoring different characteristics, such as experiments AP1, AP2, and AP10. In other cases, the datasets are independent and from different domains, such as the AP3 experiment. This case contrasts with those in which the datasets are independent but in the same application domain, with AP7.

Finally, we notice that we planned some variations of subsets that generated the experiments AP6, AP7, and AP8. In the first case, we train the CAE only with shape data sets (ShapesAll and Herring) and evaluate it in a collection of data obtained by sensors (Earthquakes). In AP7, as stated earlier, all datasets correspond to the same domain (shape). Finally, we added the set obtained by sensors in the CAE training to re-evaluate the set of time series from shapes.

All these variations are relevant to ensure a diverse analysis of the results. For example, we can easily verify whether our model can generalize well when we train it in one application domain and apply it in another.

C. Subsequence Similarity Search

The second task extends the experiments in two characteristics: the mining task and the adopted distance measure. In this scenario, we chose the subsequence similarity search under the DTW distance. We use the UCR-USP Suite tool to perform this search, and evaluate our proposal in the same datasets as the authors used in the tool's evaluation. Besides, we kept the warping window parameter as default: 5% of the query length. These datasets include time series from IMU sensors, positioning, electrical usage monitoring, and biosensors. Table II summarizes these datasets. For further information, we refer the reader to [5].

TABLE II: Summary of the datasets used in the similarity search.

ID	Dataset	Reference time series length
SS1	FoG	1,724,584
SS2	ZXY-Soccer	1,998,606
SS3	PAMAP2	3,657,033
SS4	MITDB	27,950,000
SS5	REFIT	78,596,631
SS6	PPG	333,570,000

Each of these datasets comprises a reference time series T and five queries Q_i, $i \in 1, \dots, 5$. To evaluate our proposal, we used only the T series for CAE training. All queries have 1024 observations, so we use that value as the size of the neural network's input. To adapt T for this input, we used a sliding window of length 1024 with 50% of overlap between consecutive windows. The subsequence comprised by each window becomes an input example in the CAE training phase. Figure 4 illustrates this procedure.

After the training phase, we transform T to match the new representation. For this, we again use sliding windows. However, the sliding strategy has no overlapping windows at this stage, to avoid replicating the information in the subsequences within these overlaps. Storing the encoded reference time series ends the offline phase of the experiment.

In the evaluation phase, we encode each query by the previously learned CAE. Once we train the model to have an input size equal to the length of the query, we extract the reduced representation for the query directly from the CAE, with no need for windowing.

With this information in hand, we compared the similarity search using the T and each of the queries Q_i against the same procedure on their respective reduced representations.

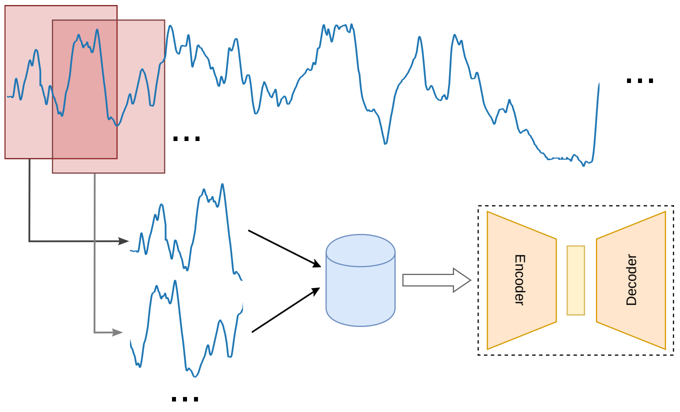


Fig. 4: Offline phase for subsequence similarity search. We used a sliding window with overlap over the reference time series to generate the input data for training the CAE.

D. Matrix Profile

The Matrix Profile is the result of the nearest neighbor join operation. From a practical standpoint, it corresponds to a sequence of subsequences similarity searches. When we use ED to compare the subsequences, we can use a convolution-based algorithm to speed up the operation. Using it with other speedup strategies leads us to perform the MP calculation orders of magnitude faster than a straightforward algorithm.

We evaluate how applying CAE impacts the runtime in this scenario. For this, we use a non-parallel implementation of STOMP. So, we can measure the direct contribution of our proposal to the algorithm. In the raw time series, we used a subsequence length of 1024. As the CAE reduces the original time series eight times, we set this parameter value as 128 in this case.

The training setup is similar to the one we applied in the similarity search task. However, as our test set is not a query, but another long time series, we also use a sliding window in the test series. Our evaluation comprises four datasets, summarized in Table III.

TABLE III: Summary of the datasets used in the matrix profile calculation.

ID	Dataset	Test time series length
MP1	Dodgers	10080
MP2	PAMAP2	501269
MP3	ZXY-Soccer	171227
MP4	mHealth	1215745

These datasets are available in the UCI Machine Learning Repository [32]. The only exception is ZXY-Soccer data [33]. We chose these datasets aiming at diversity in scale and application domain. So, we have accelerometer signals (PAMAP2), trajectories (ZXY-Soccer), loop sensors data (Dodgers), and biosignals (mHealth).

In the MP1 experiment, the training data comprises the first 20 weeks of observation, and the test data is the last five weeks of data collection.

In the MP2 experiment, we used the x-axis accelerometer observations. The training data comprises the first six subjects, and the test data is the time series collected from the last two subjects concatenated.

In MP3, we used the x coordinate of the trajectories. The training data comprises the data from the players that have more than 50000 observations in the first recorded match. The test data is composed of the time series collected from the same players in the first half of the second game, concatenated.

Finally, in MP4, we use the ECG data comprised by mHealth. To compose the training data, we concatenate the lead 1 ECG of all the subjects. The test data is the lead 2 ECG of all the subjects.

V. Results and Discussion

Except for the UCR-USP Suite source code – implemented in C++ – we perform all other steps with our Python's implementations. The advantages of implementing our solution and experiments from scratch are two-fold. We ensure that the similarity calculation is not the bottleneck of the application due to possible specific points in third-party implementations that are not the focus of this work. Besides, it gives us greater control of the execution time of each step so that we can perform the fairest evaluation of the proposal. We note that all source codes are available in this paper's online repository[2].

To evaluate our proposal, we consider two aspects: runtime improvement and preservation of the distance relationships between the series. For the first characteristic, we measure each task's execution time with and without applying CAE. Each evaluation has a different procedure, but they all have in common that only encoding and distance calculation composes the assessed runtime. For example, we do not consider the time to load the dataset as a part of the execution time. We report the speedup ratio as defined by Equation 3.

$$speedup = \frac{runtime(original\ data)}{runtime(encoded\ data)} \tag{3}$$

To assess the preservation of similarity relationships, we used both visual analysis and quantitative measures. As the different tasks have different notions about the distance space, we describe the evaluation procedure in the subsection designed for each one.

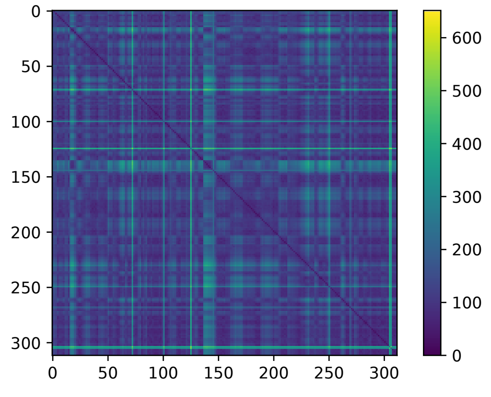
For the sake of space limitations, we do not present all the results graphically. Instead, we illustrate some of our arguments with specific examples. Besides, we provide all the plots and figures in the paper's online repository.

A. All-Pairwise Distance

We begin this section by graphically demonstrating the effect of using the CAE in all-pairwise distances calculation in the time series domain. The Figure 5 shows a distance matrix between all pairs of time series in the PigCVP dataset before and after applying the CAE.

Since it is not possible, by visual analysis, to obtain a measure of quantitative evaluation, we compare the matrices

[2]https://github.com/Tyred/BigData


(a)

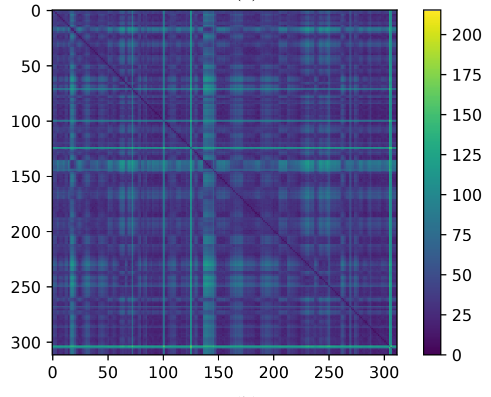

(b)

Fig. 5: All-pairwise Euclidean Distance in the original *(a)* and encoded *(b)* PigCVP dataset.

using Pearson's correlation. The use of correlation instead of some distance measure based on the values in the matrix avoids potential distortions in analysis caused by differences in scale due to different time series lengths in each case.

The Table IV features the comparative results between the distance matrices obtained and the time needed for their calculation. As this experiment does not have large scale datasets, we repeat each experiment three times and report the average runtime to avoid distortion caused by any floating. We emphasize that in order to calculate the execution time, we do not consider the reading time of the dataset in any of the cases. For the original series, we only count the pairwise Euclidean Distance calculation time. For the coded series, we count both times to transform the series using the CAE and the pairwise Euclidean Distance.

In most cases, it is possible to notice that the relationship between the distances does not change, resulting in a high correlation between the matrices. However, the correlation in the AP6 experiment was much lower than the other results. Similarly, the CAE use reduced the task runtime in all cases, except in the AP6 experiment.

An important point to notice is the comparison between the AP6, AP7, and AP8 experiments. In the first, the Pearson coefficient obtained was pretty low. On the other hand, the correlation between the matrices in the AP8 experiment was

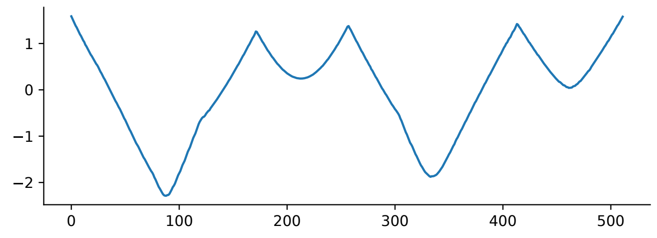
TABLE IV: Time (in seconds) to calculate the all-pairwise Euclidean distance, the relative speedup, and the Pearson coefficient obtained in each evaluation scenario.

	Runtime (original)	Runtime (encoded)	Speedup	Pearson coefficient
AP1	0.05	0.04	1.19	0.98
AP2	0.17	0.06	2.65	0.67
AP3	28.03	2.72	10.30	0.98
AP4	3.21	0.39	8.33	0.91
AP5	0.10	0.05	2.15	0.70
AP6	0.03	0.03	0.88	0.17
AP7	0.18	0.06	2.84	0.88
AP8	0.18	0.06	2.90	0.93
AP9	2.78	0.39	7.16	0.74
AP10	1.63	0.28	5.89	0.90

higher than the AP7 experiment. This shows that, although the Herring and ShapesAll datasets do not have enough information to train a CAE capable of learning good representations of the Earthquakes dataset, the last dataset, even though it is from another application domain, helps to achieve a CAE with better capability to reduce the time series of the ShapesAll dataset.

Our hypothesis for this phenomenon is that this is because the datasets used as training in the AP6 experiment have much less complexity than Earthquakes. Batista et al. noted that the complexity of a time series might significantly impact time series mining algorithms. Training using less complicated series was not beneficial to induce an encoder for more complicated series. In contrast, training the CAE using more complex data seems to generalize the model.

For the sake of clarification, Figure 6 shows a time series of the Earthquake dataset and one from the ShapesAll dataset. Note the difference between the number of peaks and valleys and the number of trend changes between the two graphs.


(a)

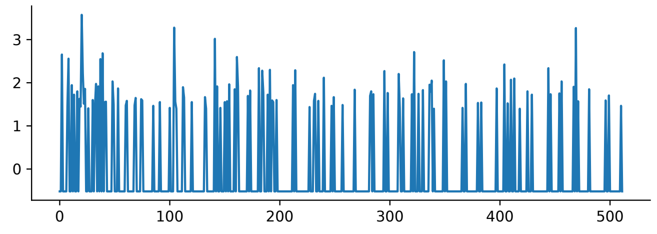

(b)

Fig. 6: Examples of time series from the ShapesAll *(a)* and Earthquakes *(b)* datasets.

Besides, the quality of the CAE may be affected by the small number of training examples. For example, there are only 724 training examples available for the AP2 experiment.

From the runtimes observed, we recognize a limitation of this experiment. Because we restrict ourselves with datasets with time series of the same length, the datasets contain few examples. However, this experiment demonstrates that reducing dimensionality with the CAE has excellent potential for reducing the runtime in the calculation of similarity. This potential is even more evident when we notice a proportionality trend between the time taken to calculate the distances and the speedup achieved by our proposal, as illustrated in Figure 7. In other words, the use of the CAE shows a trend that the longer the time needed to calculate the distances between the time series, the greater the advantage of using the autoencoder as a speedup tool.

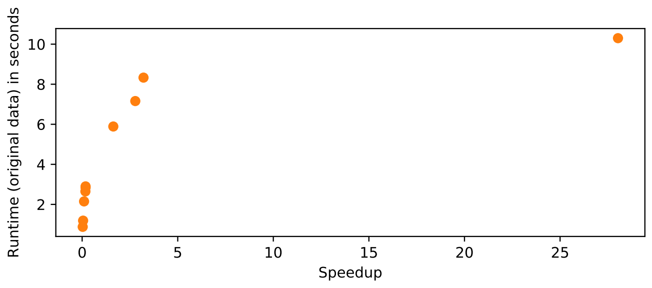


Fig. 7: Speedup ratio vs. runtime in the original series in the all-pairwise ED calculation (each bullet point in the chart represents one dataset).

B. Subsequence Similarity Search

In the second experiment, the datasets used are larger, and we use a more expensive distance measure. In this scenario, we expect a significant reduction in execution time when we apply dimensionality reduction. Table V summarizes the results obtained, presenting the average of the execution times and speedup ratios among all queries of each dataset.

TABLE V: Average time (in seconds) to calculate the DTW-based similarity search and the relative average speedup for each dataset.

ID - Dataset	Runtime (original)	Runtime (encoded)	Speedup
SS1 - FoG	271.59	0.51	456.19
SS2 - ZXY-Soccer	4.22	0.10	42.82
SS3 - PAMAP2	6.40	0.32	18.68
SS4 - MITDB	139.01	2.76	54.14
SS5 - REFIT	77.48	4.25	18.32
SS6 - PPG	4272.84	369.54	11.20

Analyzing these results, it is impossible to observe a direct relationship between the lengths of the reference series or the runtime in the original data with the obtained speedup ratio. However, there seems to be a direct relationship with the difficulty of indexing methods in pruning the DTW distance calculation. Consider the cases SS3 and SS5, which obtained similar speedup ratios. The UCR-USP Suite was able to prune 99.01% and 98.88% of the DTW calculations in these datasets, respectively.

On the other hand, the best result we obtained refers to the SS1 experiment. For the first query of this dataset, it was necessary to calculate the DTW distance for 58.93% of the substrings. When we search in the reduced time series domain, that value changes to 20.29%.

To verify CAE's effect on preserving similarities, we compare the query with its respective nearest neighbors in the original series and when CAE is applied. Figures 8, 9, 10, 11, 12, and 13 , present these (z-normalized) subsequences for the first query of the datasets FoG, ZXY-Soccer, PAMAP2, MITDB, REFIT and PPG, respectively. We note that the time series that represents the nearest neighbor found using CAE is not coded in the figures. To obtain it, we access the original series in the equivalent position.

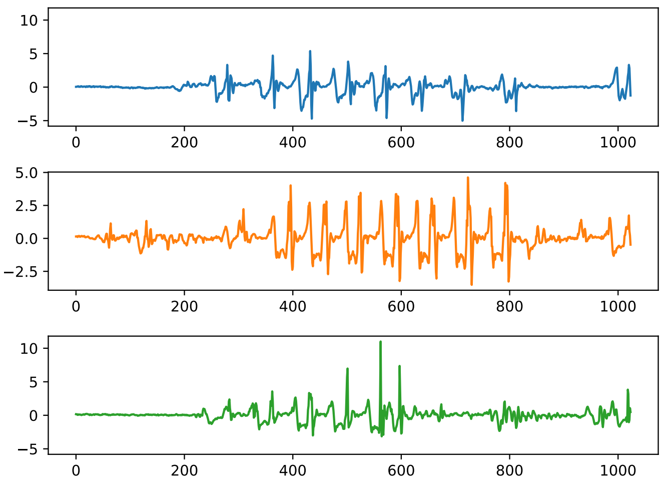


Fig. 8: First query of the dataset FoG *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*

In none of the cases analyzed, the nearest neighbor with and without applying the CAE was the same subsequence. However, it is possible to notice in these figures that the subsequences returned using CAE are very similar to the queries. In some cases, the encoded series seems to add robustness to the search. For example, the narrow valleys at the beginning of the REFIT dataset's query (Figure 12) are more likely to be noise than a low plateau, such as in the neighbor obtained in the original series.

C. Matrix Profile

Finally, we present and discuss the results of our last experiment. Once again, we provide runtime and speedup ratios. Table VI shows these results.

This experiment evidences a trend that we noticed in the all-pairwise distance experiment, that the larger the data set, the more significant the speedup achieved by our proposal. In the largest of the data sets analyzed in this task, MP4, our method

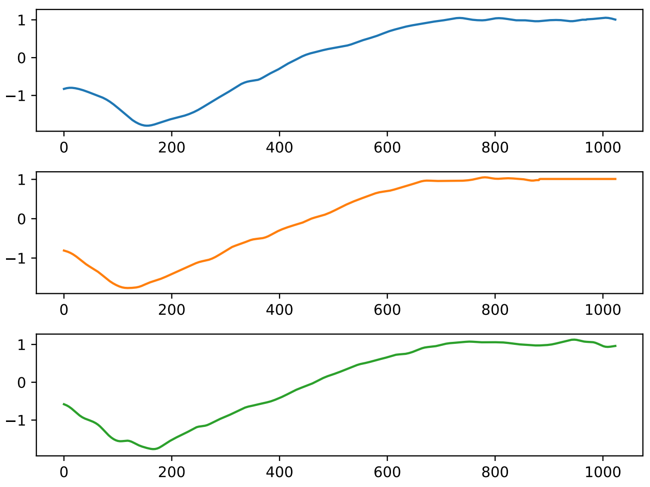


Fig. 9: First query of the dataset ZXY-Soccer *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*

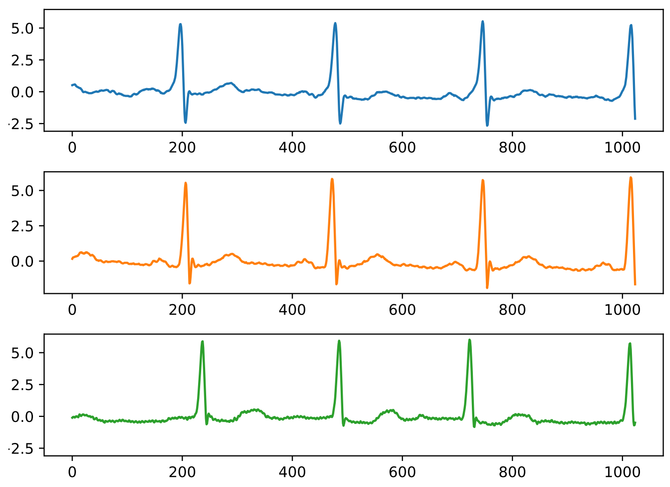


Fig. 11: First query of the dataset MITDB *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*

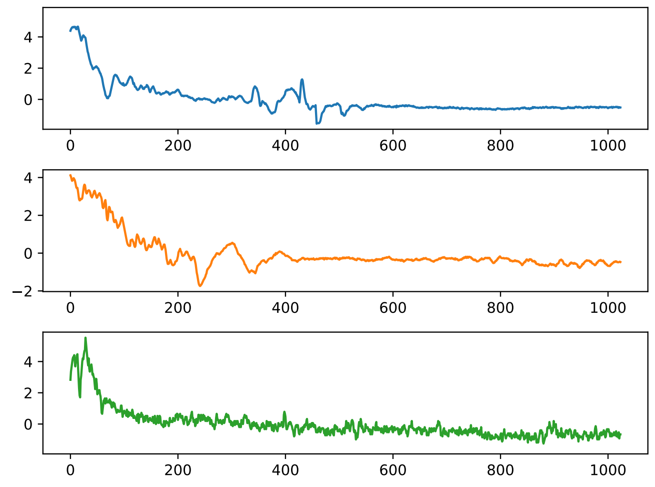


Fig. 10: First query of the dataset PAMAP2 *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*

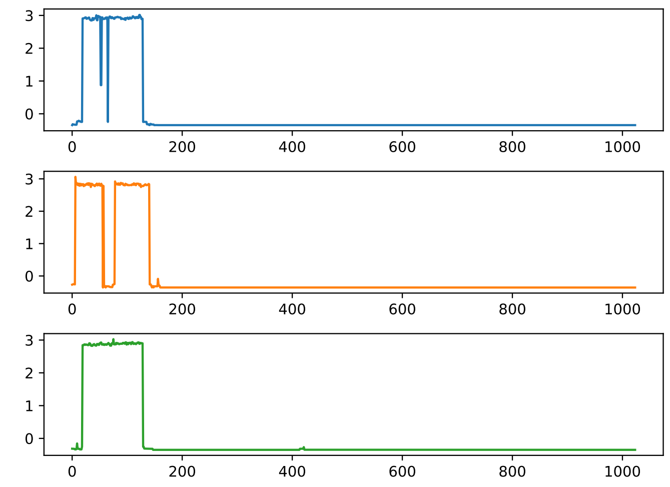


Fig. 12: First query of the dataset REFIT *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*

is two orders of magnitude faster than calculation using the raw time series.

To analyze the preservation of distance relations, we observed the best-pair motifs obtained from MP: the subsequences that have the nearest neighbor with the smallest distance among all the subsequence pairs [34]. We note that

TABLE VI: Time (in seconds) to calculate the MP and the relative speedup for each dataset.

	Runtime (original)	Runtime (encoded)	Speedup
MP1 - Dodgers	0.86	0.11	7.89
MP2 - PAMAP2	261.18	3.41	76.54
MP3 - ZXY-Soccer	2174.84	26.19	83.04
MP4 - mHealth	18285.62	181.26	100.88

the motif found in the MP1 and MP3 is the same in both raw and encoded data. In other words, the index where the subsequence considered motif in the domain of the original series starts is equal[3] to the equivalent index of that obtained using the CAE.

For the other datasets, we make use of visual inspection. Figure 14 shows the best-pair motifs found with both evaluated strategies in the Dodgers dataset.

Although they are not the same pairs of subsequences, we note that these motifs have similar shapes in both scenarios. That is, both pairs represent meaningful motifs. Furthermore,

[3]Given that the CAE reduces the series length by eight times, we consider that a difference of up to eight positions for the beginning of the subsequence is negligible.

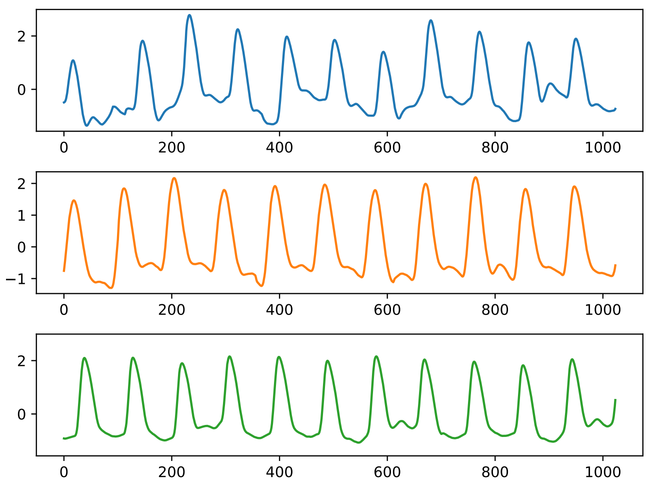
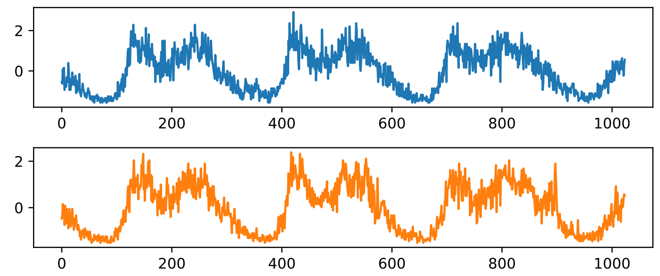
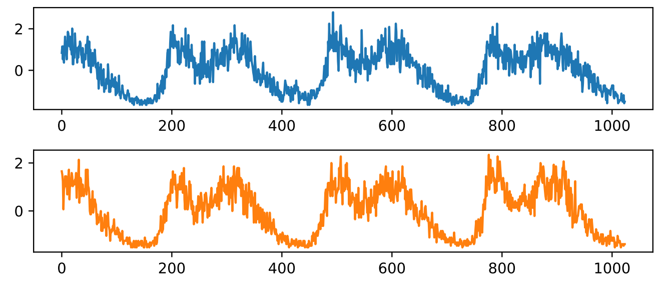


Fig. 13: First query of the dataset PPG *(top, blue) and its nearest neighbor in the original representation (center, orange) and using CAE (bottom, green).*



(a)



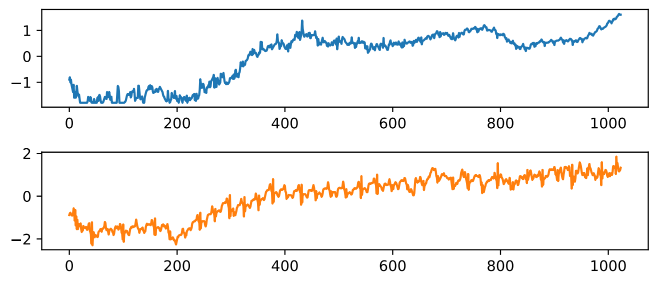
(b)

Fig. 14: Best pair of motifs found in the raw *(a)* and encoded *(b)* time series in the Dodgers dataset.

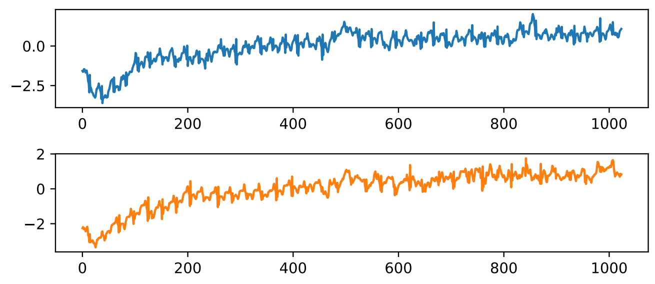
the motif pairs found with and without the use of CAE have apparent similarities between them.

Figure 15 presentes the motifs discovered in the mHealth data.

As in the previous case, both pairs of subsequences seem to represent significant motifs in this case. Additionally, this result has a peculiarity. Perhaps the reader may perceive that the motifs found with the use of CAE look more than the other. While the Euclidean distance between the subsequences



(a)



(b)

Fig. 15: Best pair of motifs found in the raw *(a)* and encoded *(b)* time series in the mHealth dataset.

is smaller for the motifs discovered in the raw series, the DTW distance is smaller between the other pair. This fact probably occurs because Euclidean distance is more likely to match less complex series [35]. However, our method seems to add robustness against this effect.

VI. Final Remarks

With the growing volume of temporal data, we believe that new methods should be proposed to guarantee similarity-based time series mining algorithms' scalability. This thought is shared among researchers and, therefore, several tools have been presented for this purpose.

We proposed an approach based on convolutional autoencoders, which is orthogonal to state-of-the-art techniques in time series mining. We have shown empirically that the use of CAE consistently improves the efficiency of these methods. Besides, we noticed that the greater the difficulty of these techniques in improving the execution time, the greater our proposal's benefit.

Since the dimensionality reduction can cause distortions in the obtained results, we also evaluate relative distances' preservation. In some cases, unfortunately, we notice significant distortions. However, these cases are exceptions and they are associated with little data for training and notable differences in complexity between time series. When we train a CAE with time series with simple shapes and apply it to complex series, the result is not satisfactory. Nevertheless, it does not stand in the opposite situation.

In general, we believe that using CAE can be very beneficial for practical applications that require the calculation of similarity between time series. However, it only holds if there exists sufficient data for offline training, with characteristics similar to those expected for new data. Since the autoencoder training is unsupervised, it is enough that these data are collected or simulated in some way, without the need for labeling.

As future work, we intend to evaluate our proposal's effects on mining tasks such as classification and clustering. Also, we will extend the method for dealing with multidimensional series.

Acknowledgment

We would like to acknowledge that this research was funded by grants #2017/24340-6 and #2019/06080-2, São Paulo Research Foundation (FAPESP).

References

[1] M. Khan, X. Wu, X. Xu, and W. Dou, "Big data challenges and opportunities in the hype of industry 4.0," in *IEEE International Conference on Communications*. IEEE, 2017, pp. 1–6.

[2] M. Tahmassebpour, "A new method for time-series big data effective storage," *IEEE Access*, vol. 5, pp. 10 694–10 699, 2017.

[3] X. Wang, A. Mueen, H. Ding, G. Trajcevski, P. Scheuermann, and E. Keogh, "Experimental comparison of representation methods and distance measures for time series data," *Data Mining and Knowledge Discovery*, vol. 26, no. 2, pp. 275–309, 2013.

[4] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping," *ACM Transactions on Knowledge Discovery from Data*, vol. 7, no. 3, pp. 1–31, 2013.

[5] D. F. Silva, R. Giusti, E. Keogh, and G. E. A. P. A. Batista, "Speeding up similarity search under dynamic time warping by pruning unpromising alignments," *Data Mining and Knowledge Discovery*, vol. 32, no. 4, pp. 988–1016, 2018.

[6] M. Linardi, Y. Zhu, T. Palpanas, and E. Keogh, "Matrix profile goes mad: variable-length motif and discord discovery in data series," *Data Mining and Knowledge Discovery*, 2020.

[7] R. Giusti and G. E. Batista, "An empirical comparison of dissimilarity measures for time series classification," in *Brazilian Conference on Intelligent Systems*. IEEE, 2013, pp. 82–88.

[8] T. W. Liao, "Clustering of time series data—a survey," *Pattern recognition*, vol. 38, no. 11, pp. 1857–1874, 2005.

[9] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565–592, 2015.

[10] E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping for data mining applications," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2000, pp. 285–289.

[11] H. Sakoe and S. Chiba, "Dynamic programming algorithm optimization for spoken word recognition," *IEEE transactions on acoustics, speech, and signal processing*, vol. 26, no. 1, pp. 43–49, 1978.

[12] E. Keogh and C. A. Ratanamahatana, "Exact indexing of dynamic time warping," *Knowledge and Information Systems*, vol. 7, no. 3, pp. 358–386, 2005.

[13] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile i: all pairs similarity joins for time series: a unifying view that includes motifs, discords and shapelets," in *IEEE International Conference on Data Mining*. IEEE, 2016, pp. 1317–1322.

[14] C. Guo, H. Li, and D. Pan, "An improved piecewise aggregate approximation based on statistical features for time series mining," in *International Conference on Knowledge Science, Engineering and Management*. Springer, 2010, pp. 234–244.

[15] K. Sirisambhand and C. A. Ratanamahatana, "A dimensionality reduction technique for time series classification using additive representation," in *Third International Congress on Information and Communication Technology*. Springer, 2019, pp. 717–724.

[16] S. Salvador and P. Chan, "Toward accurate dynamic time warping in linear time and space," *Intelligent Data Analysis*, vol. 11, no. 5, pp. 561–580, 2007.

[17] R. Wu and E. Keogh, "Fastdtw is approximate and generally slower than the algorithm it approximates," *arXiv preprint arXiv:2003.11246*, 2020.

[18] T. Rakthanmanon, B. Campana, A. Mueen, G. E. A. P. A. Batista, B. Westover, Q. Zhu, J. Zakaria, and E. Keogh, "Searching and mining trillions of time series subsequences under dynamic time warping," in *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 262–270.

[19] D. F. Silva and G. E. A. P. A. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in *SIAM International Conference on Data Mining*. SIAM, 2016, pp. 837–845.

[20] G. Al-Naymat, S. Chawla, and J. Taheri, "Sparsedtw: a novel approach to speed up dynamic time warping," in *Australasian Data Mining Conference*, 2009, pp. 117–127.

[21] Y. Zhu, S. Gharghabi, D. F. Silva, H. A. Dau, C.-C. M. Yeh, N. S. Senobari, A. Almaslukh, K. Kamgar, Z. Zimmerman, and G. Funning, "The swiss army knife of time series data mining: ten useful things you can do with the matrix profile and ten lines of code," *Data Mining and Knowledge Discovery*, 2020.

[22] A. Mueen, Y. Zhu, M. Yeh, K. Kamgar, K. Viswanathan, C. Gupta, and E. Keogh, "The fastest similarity search algorithm for time series subsequences under euclidean distance," August 2017, http://www.cs.unm.edu/~mueen/FastestSimilaritySearch.html.

[23] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra, "Dimensionality reduction for fast similarity search in large time series databases," *Knowledge and information Systems*, vol. 3, no. 3, pp. 263–286, 2001.

[24] Y. Zhu, C.-C. M. Yeh, Z. Zimmerman, and E. Keogh, "Matrix profile xvii: Indexing the matrix profile to allow arbitrary range queries," in *IEEE International Conference on Data Engineering*. IEEE, 2020, pp. 1846–1849.

[25] Y. Wang, H. Yao, and S. Zhao, "Auto-encoder based dimensionality reduction," *Neurocomputing*, vol. 184, pp. 232–242, 2016.

[26] M. Ali, M. W. Jones, X. Xie, and M. Williams, "Timecluster: dimension reduction applied to temporal data for visual analytics," *The Visual Computer*, vol. 35, no. 6-8, pp. 1013–1026, 2019.

[27] T. Kieu, B. Yang, and C. S. Jensen, "Outlier detection for multidimensional time series using deep neural networks," in *IEEE International Conference on Mobile Data Management*. IEEE, 2018, pp. 125–134.

[28] P. Ramachandran, B. Zoph, and Q. V. Le, "Swish: a self-gated activation function," *arXiv preprint arXiv:1710.05941*, vol. 7, 2017.

[29] P. Singh and A. Manure, "Introduction to tensorflow 2.0," in *Learn TensorFlow 2.0*. Springer, 2020, pp. 1–24.

[30] R. J. Kate, "Using dynamic time warping distances as features for improved time series classification," *Data Mining and Knowledge Discovery*, vol. 30, no. 2, pp. 283–312, 2016.

[31] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, 2019.

[32] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[33] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen, "Soccer video and player position dataset," in *ACM Multimedia Systems Conference*, 2014, pp. 18–23.

[34] A. Mueen, "Time series motif discovery: dimensions and applications," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 4, no. 2, pp. 152–159, 2014.

[35] G. E. A. P. A. Batista, E. J. Keogh, O. M. Tataw, and V. M. A. De Souza, "Cid: an efficient complexity-invariant distance for time series," *Data Mining and Knowledge Discovery*, vol. 28, no. 3, pp. 634–669, 2014.