Enhance Robustness of Deep-Hashing Model in Streaming Time-Series via Ambiguous-Sample Awareness Training Framework

Richard Tapia[1], **Yifeng Gao**[1]

[1]University of Texas Rio Grande Valley
richard.tapia01@utrgv.edu, yifeng.gao@utrgv.edu

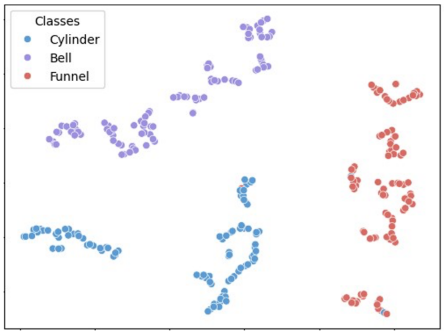
Abstract

Recently, deep-learning based hash models aim to build deep hash functions to effectively index the data based on their semantic meaning instead of purely similarity, as it receives a large amount of attention. However, few studies have studied the deep-hashing model in the streaming decision-making system for time series data, in which the model needs to perform tasks after every fixed time stamp. One of the challenges that the model will face is the ambiguous samples — samples that are similar to the labeled data due to overlapping but not associate with the same meaning. In this paper, we introduce an ambiguous-sample awareness deep hashing training framework to address this challenge. Specifically, we proposed a novel loss named Ditanciation Loss. The loss especially encourages the model to separate the semantically ambiguous samples to enhance the robustness of the deep hashing model. The results of the experiment demonstrate that the proposed method could enhance the mAP performance when the system is deployed into streaming setting.

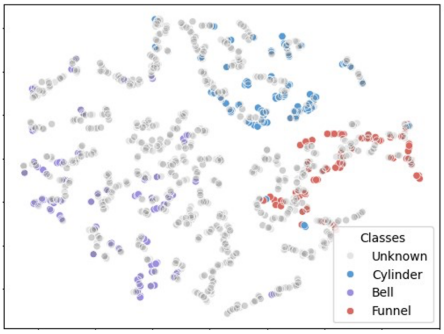
Introduction

Time series data is one of the most broad data types that exist, and that is studied by diverse fields such as finance (Tsay 2005; Sezer, Gudelek, and Ozbayoglu 2020; Kim 2003), healthcare (Chuah and Fu 2007; Pyakillya, Kazachenko, and Mikhailovsky 2017), and meteorology (Hewage et al. 2020; Karevan and Suykens 2020). With the advancement of sensor technology, large-scale time series data is widespread. Among all these applications, retrieving semantically similar time series data plays an important role in a wide range of applications (Camerra et al. 2010; Palpanas 2020; Yagoubi et al. 2017). Recently, there has been a growth in studies interested on using deep-learning based hash model (Cao et al. 2017; Su et al. 2018; Zhu et al. 2016) to effectively index the data based on complex semantic meaning and allowing enhanced search and retrieval processes. These models often consider a set of time series and its corresponding label, where the aim is to generate a hashcode such that if the shared similar label, their hash code should be close.

While many time series deep hashing work have been proposed in the recent years (Cao et al. 2017; Su et al. 2018; Zhu et al. 2016), very few studies have explored deep streaming hashing systems, which the hashing task is executed in every fixed time span. Such setting is widely used and exists in many applications. For example, considering a system that analyzes power demand pattern behavior (Egarter and Elmenreich 2015; Müller et al. 2017), it requires the system to check the existence of the pattern in every time stamp since the activity can appear in arbitrary time. The unique challenge in this setting is the highly similar but not semantically similar samples that may exist and mislead the deep hashing model during the training.



(a) Instance-wise Hashing



(b) Streaming Hashing

Figure 1: **(a)** t-SNE Visualization of well segmented and labeled time series instances **(b)** t-SNE Visualization in streaming hashing cases, a set of segments with no meaningful associated label occurred. And can mixed with meaningful data easily

To demonstrate the severity of the challenge caused by streaming time series, an example is illustrated in Figure 1. Figure 1.a illustrates the t-SNE visualization of each sam-

ples when the data is well segmented (e.g. instance-wise hashing). In this case, by visualizing the t-SNE of the raw time-series data segments, we could find that three classes of samples are well separated. Thus, it is much easier to build deep hashing model. However, in the streaming setting, the t-SNE visualization of all the samples is shown in Figure 1.b. In this case, large amount of "ambiguous" data that represents misaligned or overlapping segments exist, and impact model performance. These samples are a partial combination of data across instances and do not have an accurate meaning in most cases. On the other hand, they are also close to the labeled data, which make them hard to distinguish (highlighted in gray, a considerable amount of sample points are very similar to the actual label data). Such samples significantly increase the complexity of the embedding spaces.

In this paper, we propose a novel training framework by specially considering separating the semantically ambiguous samples to enhance the robustness of the deep hashing model against the ambiguous samples. To solve this problem, extensive experiments were conducted to test the proposed method against a large variety of time-series data. The result demonstrates that the proposed method could enhance the mAP performance when the system is deployed into a streaming setting.

Related Work

Hashing techniques have been widely used in a wide range of time series data mining tasks such as motif discovery (Gao and Lin 2017, 2019; Li et al. 2015), anomaly detection (Keogh, Lin, and Fu 2005; Mercer and Keogh 2024), and rule discovery (Shokoohi-Yekta et al. 2015). In these existing frameworks, a hashing function (typically based on data similarity), is applied to perform constant time complexity approximate time series retrieval. The hash function is often applied to all the segments of a fixed or variable length extracted from one long streaming time series.

Recently, various supervised deep-learning based hash models have been proposed to design hashing functions that could perform fast nearest neighbor retrieval of high-dimensional data based on semantic similarity (Cao et al. 2017; Su et al. 2018; Zhu et al. 2016), induced through labels. Several time series deep hashing frameworks have been proposed (Song et al. 2018; Wang and Palpanas 2023). Song et al. introduced a deep rth root based loss function to transform time series into binary embedding for the task of information retrieval. Wang et al. introduces a framework called SeaNet (Wang and Palpanas 2023) for unsupervised deep hashing with the guidance of symbolic representation.

All of these proposed work are not considering the streaming hashing setting mentioned above, where ambiguous samples in streaming data arise due to the existence of overlapping segments.

Our work proposes a special hashing loss that is aware of the semantically ambiguous samples, and is orthogonal to the existing research.

Methodology

Problem Definition

Extracting from a streaming time series T via a sliding window of length L, we denote N labeled time-series data as $X^n \in \mathbb{R}^{N \times L \times D}$, and $x_i \in X$ represents a time series data with L length and D dimension. We also denote the corresponding labels as $y_i \in \{0, \ldots, C-1\}$ where C denotes the total number of classes. Additionally, we denote time series $X^u \in \mathbb{R}^{N' \times L \times D}$ belonging to the ambiguous samples (ambiguous subsequences due to overlapping). Similarly, we denote the i_{th} sample belonging to X^u as x_i^u.

Our goal is to obtain a deep hashing model, such that given an input time series $x_i' \in X^n \cup X^u$, it can produce a binary code $b_i \in \{-1, 1\}^d$, typically obtained through applying $sign$ function in the embedding features obtained by a deep learning model $h(.)$ such that the time series that belongs to the same class will have a higher chance of obtaining similar binary code and if b_i is generated from time series in X^n, it should be distinguishable compared to binary code generated from time series in X^u, minimizing the impact of irrelevant or ambiguous signals. In a real-world scenario, where the data is constantly flowing in, it is unavoidable to receive the ambiguous or meaningless type of data. Thus, it is important to identify and remove such ambiguous samples while maintaining normal hashing performance.

Proposed Method

Overall framework

The overall framework of the proposed method is shown in Figure 2. The framework consists of two components. First, the embedding features and the are computed based on the input time series and their label, from the embedding we then compute the class-embedding centric μ (Figure 2.top). Then the proposed loss is applied to train the model to enhance the separation (Figure 2.bottom). Overall, the proposed loss aims to minimize the impact of ambiguous samples via the Distanciation Loss $\mathcal{L}^u$ and enhance hashing performance via a prototypical loss $\mathcal{L}^c$.

Compute embedding features and the class-embedding centric

Given a time series dataset X, the framework first applies an encoder model $h(.)$ and obtains their corresponding embedding features:

$$E = h(X) \tag{1}$$

where $e_i \in E$ represents the embedding of $x_i \in X$. In the experiment, the encoder $h(.)$ is the ResNet34 model with 1-D Convolution layer.

Next, we map the embedding into hypersphere space $e_i' \in \mathcal{S}^{d-1}$ to minimize the information loss caused by discretization introduced by the $sign(.)$ function:

$$e_i' = \frac{e_i}{||e_i||_2} \tag{2}$$

Finally, the embedding centric for class k, μ_k, in hypersphere $\mathcal{S}^{d-1}$ of each class of time series is calculated by:

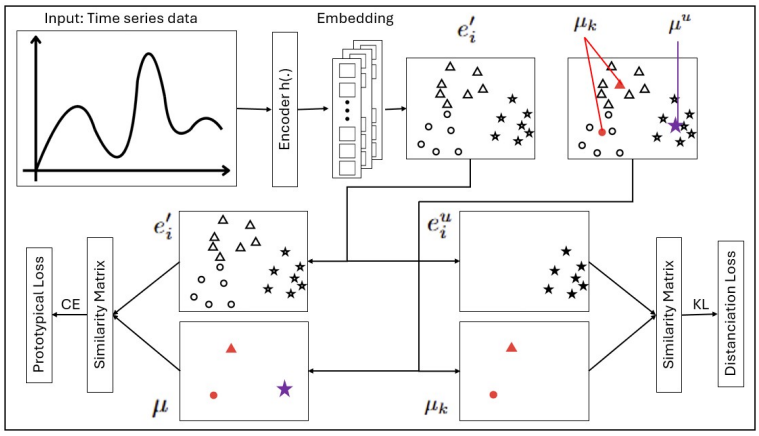


Figure 2: Overall framework for proposed method

$$\mu_k = \frac{1}{N_k} \sum_{y_i \in k} e'_i \tag{3}$$

where N_k denotes total number of data that belongs to class k. We note $\mu^n = \{\mu_k | k = 0, 1, \ldots, C-1\}$ as all the class centric.

Similarly, we obtain the centric for the ambiguous data via:

$$\mu^u = \frac{1}{N_u} \sum_{x_i \in X^u} e'_i \tag{4}$$

where N_u denotes total number of ambiguous data. We denote $\mu = \mu^n \cup \mu^u$ as all the centric we have in the datasets.

Proposed Loss

Given centric μ and embedding e'_i, we next compute our proposed loss. The loss consists of two components. A prototypical loss $\mathcal{L}_c$ to measure the separation between each centric.

Prototypical Loss Given the embedding e'_i and all the centric μ. We predict $p(\hat{y}_i = k|x_i)$ through similarity against all centric, including the ambiguous data:

$$p(\hat{y}_i = k|e'_i) = \frac{\exp(sim(e'_i, \mu_k))}{\sum_{\mu' \in \mu} \exp(sim(e'_i, \mu'))} \tag{5}$$

where $sim(e'_i, \mu_k)$ denotes cosine similarity between e'_i and μ_k. Because e'_i and μ_k both belong to $\mathcal{S}^{d-1}$, the cosine similarity can be computed via inner product (e.g. $sim(x, y) = x^T y$). Next, we compute the chance that the data belong to X^u via:

$$p(x_i \in X^u | e'_i) = 1 - \sum_{k=0}^{C-1} p(\hat{y}_i = k|e'_i) \tag{6}$$

The proposed loss is computed via the cross entropy function against the ground truth labels:

$$\begin{aligned} \mathcal{L}^c_i = &- \mathbf{1}\{x_i \in X^n\} \log p(\hat{y}_i = y_i|e'_i) \\ &- \mathbf{1}\{x_i \in X^u\} \log p(x_i \in X^u|e'_i) \end{aligned} \tag{7}$$

And the loss $\mathcal{L}^C$ is:

$$\mathcal{L}^c = \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}^c_i \tag{8}$$

Distanciation Loss $\mathcal{L}^u$ Next, we introduce the distanciation loss. Intuitively, we should expect the ambiguous samples to be far away with any samples that have an associated label. To achieve this goal, we define the following terms, let e^u_i be the embedding generated from $x_i \in X^u$ and let $p(\hat{y} = k|e^u_i)$ be the probability that the model predict the data belong to class k given e^u_i via:

$$p(\hat{y} = k|e^u_i) = \frac{\exp(sim(e^u_i, \mu_k))}{\sum_{\mu' \in \mu_\parallel} \exp(sim(e^u_i, \mu'))} \tag{9}$$

Since e^u_i is encoded from $x_i \in X^u$, we expect the output after processing the embedding to be $p(\hat{y} = k|e^u_i)$ as a uniform distribution (e.g. has same probability across all classes). Thus, we define our distanciation loss as the KL divergence between $p(\hat{y} = k|e^u_i)$ and the uniform distribution across the classes:

$$\begin{aligned} \mathcal{L}^u &= D_{KL}(p \,\|\, \mathcal{U}_C) \\ &= \sum_{e^u_i \in X^u} \sum_{k=0}^{C-1} -\frac{1}{C} \log\left(C \cdot p(\hat{y}^u = k|e^u_i)\right) \end{aligned} \tag{10}$$

where $\mathcal{U}_C$ denotes the uniform distribution over C classes.

The Distanciation Loss reduces the likelihood that ambiguous samples are assigned to a specific class by encouraging their predicted probabilities to be uniformly distributed. This ensures ambiguous samples do not cluster near class centroids but remain equally distant from all classes.

Overall, our final proposed loss is a weighted sum of the two losses:

$$\mathcal{L} = \mathcal{L}^c + \lambda \mathcal{L}^u$$

where λ balances the Distanciation Loss and Prototypical Loss.

Experiment

In this section, we demonstrate that the proposed loss can have better performance in term of handling the ambiguous samples. Throughout the experiments, we use ResNet34 as our backbone. We set the number of bits to 64, learning rate is 0.001, and the NVIDIA L4 GPU is used in the experiment. All our experiments are conducted via the Google Cloud Platform.

Datasets

We generate our testing cases from 40 datasets from UCR Time Series Classification Dataset (Dau et al. 2019) which covers 4 different types of applications, Sensor, Motion, ECG, and Simulation, to ensure a diverse characteristic. Next, we will use these instances to generate the streaming hashing experiment.

Simulating Streaming Setting We next describe the preprocessing step to transfer the data into streaming fashion and annotated the ambiguous data given a set of annotated time series instance from UCR time series classification dataset in the experiment. We first concatenate all the time series instances together to form a long time series. Then, we use a sliding window to pass through this streaming time series and extract the data with the same length as an instance. We denote the instances that are overlapping with two known samples but are dissimilar to the all known samples as the ambiguous sample. In the experiment, we maintain the ambiguous samples size is 4 times of known sample, to simulate the cases that during streaming hashing, majority of time series will be irrelevant to the class label. All the experiments are repeated 3 times and the average performance is reported.

Baseline

Because most of deep hashing model we are using is not focused on the same setting, we chose a trivial method of using the **ambiguous samples as a class** as a baseline for us to compare our method. To do this, we only consider the prototypical loss on the data. This baseline was trained with all the data X, to better compare our method and the baseline to the real-world streaming data challenge.

Evaluation

The evaluation is performed separately for each dataset to provide a clear comparison of performance. The evaluation process involves a detailed comparison of model performance under different scenarios to assess the impact of the proposed loss function. The evaluation metric we are using it the mean Average Precision (mAP), a common metric to measure the ability of conducting information retrieval.

We compare our method to the baseline in two different perspective metric.

- **mAP of all data**: we measure the mAP performance across all testing data, this gives us an idea of how well each approach performs with the real-world type of data.

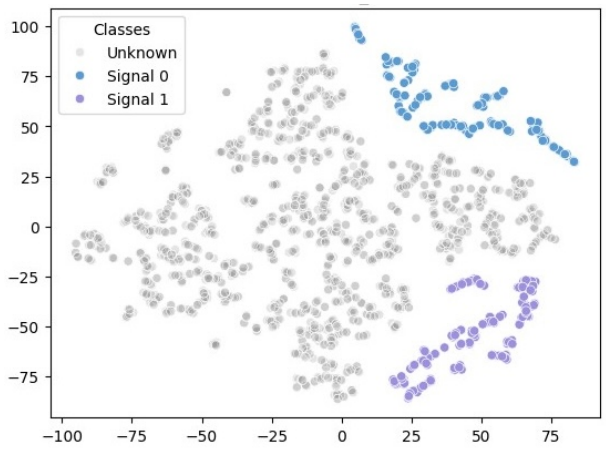


Figure 3: t-SNE Visualization of TwoLeadECG Dataset embedding with proposed loss

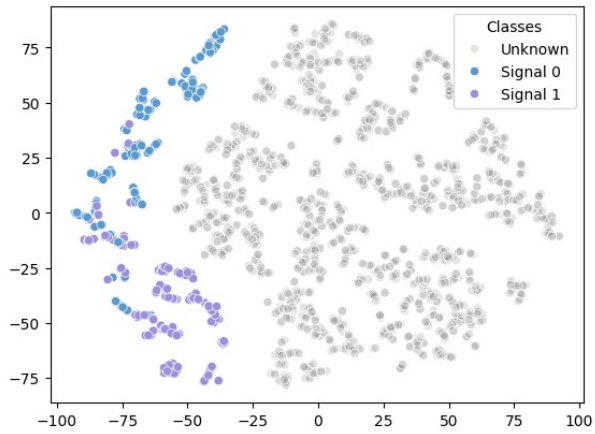


Figure 4: t-SNE Visualization of TwoLeadECG Dataset embedding ambiguous samples as class

- **mAP of labeled data** We measure the mAP values only on the labeled data X^n, this way we can see how well each approach can preserve the performance of retrieving the meaningful classes.

We expect the better method should has higher mAP in both cases.

Results Comparison

Our dataset-level comparison results is shown in Table 1, and the summary of comparison is shown in Table 2.

According to Table 1 and 2, in terms of the mAP performance across all the data including correctly distinguish the ambiguous samples, we found that our proposed method is better than the simple solution of treating the ambiguous samples as an additional class. Among all the 40 tested dataset, the proposed method is better on 23 out of 40 datasets. And the overall accuracy is higher than the original setting (0.818 vs. 0.804). This shows that the proposed loss can achieve better performance than simply treating ambigu-

Dataset	Ambiguous samples as a class (all data)	Proposed (all data)	Ambiguous samples as a class (original data)	Proposed (original data)
Computers	**0.806**	0.797	0.594	**0.623**
FreezerRegularTrain	0.884	**0.998**	0.666	**0.921**
FreezerSmallTrain	0.674	**0.907**	0.676	**0.792**
HouseTwenty	**0.782**	0.769	0.736	**0.745**
LargeKitchenAppliances	**0.763**	0.749	**0.513**	0.508
PowerCons	0.911	**0.913**	0.699	**0.791**
RefrigerationDevices	0.708	**0.773**	**0.497**	0.455
ScreenType	**0.825**	0.791	**0.462**	0.447
SmallKitchenAppliances	**0.785**	0.757	0.515	**0.591**
AtrialFibrillation	**0.721**	0.644	0.277	**0.463**
ECG200	0.957	**0.972**	**0.731**	0.724
ECG5000	0.405	**0.632**	0.690	**0.817**
ECGFiveDays	0.965	**1.000**	0.658	**0.840**
StandWalkJump	**0.719**	0.686	0.194	**0.450**
TwoLeadECG	0.936	**1.000**	0.691	**0.974**
ArticularyWordRecognition	**0.829**	0.805	**0.113**	0.108
Haptics	**0.876**	0.856	0.331	**0.348**
InlineSkate	0.845	**0.859**	**0.314**	0.273
ToeSegmentation1	0.796	**0.803**	0.673	**0.765**
ToeSegmentation2	**0.768**	0.750	0.685	**0.712**
Worms	**0.831**	0.830	0.432	**0.477**
WormsTwoClass	**0.840**	0.835	0.591	**0.616**
Car	**0.907**	0.876	0.371	**0.415**
DodgerLoopDay	0.846	**0.881**	**0.300**	0.254
DodgerLoopGame	0.903	**0.933**	0.563	**0.789**
DodgerLoopWeekend	0.967	**0.982**	**0.779**	0.755
Earthquakes	0.716	**0.741**	**0.735**	0.734
ElectricDeviceDetection	**0.306**	0.153	0.833	**0.847**
Lightning2	**0.869**	0.787	**0.686**	0.627
Lightning7	0.832	**0.841**	0.284	**0.305**
MoteStrain	**0.842**	0.791	0.679	**0.721**
Plane	0.841	**0.958**	0.392	**0.603**
SonyAIBORobotSurface1	0.992	**0.998**	0.724	**0.825**
Trace	0.920	**0.948**	0.513	**0.612**
BME	0.918	**0.996**	0.518	**0.744**
CBF	0.995	**0.998**	0.564	**0.815**
ChlorineConcentration	0.903	**0.908**	0.514	**0.555**
ShapeletSim	**0.737**	0.715	0.579	**0.602**
TwoPatterns	0.092	**0.139**	0.399	**0.470**
UMD	0.947	**0.960**	0.517	**0.663**

Table 1: Comparison of ambiguous samples as a class and proposed method on all data and original data.

Metric	Ambiguous samples as a class (all data)	Proposed (all data)	Ambiguous samples as a class (original data)	Proposed (original data)
Average	0.804	**0.818**	0.542	**0.619**
Wins	17	**23**	10	**30**

Table 2: Summary of average performance and number of wins for each method.

ous samples as one class in term of the overall performance.

Next, we evaluate the proposed method based on the mAP performance in all the labeled data. In this setting, the proposed method is significantly better than the trivial solution. Among all the 40 tested datasets, the proposed method is better on 30 out of 40 datasets. And the average mAP value is also significantly higher (0.619 vs. 0.542). The result demonstrates that the proposed method can achieve significant better performance in term of preserving the information retrieval performance in the ambiguous labeled data.

Overall, the can conclude that by introducing the new loss function, the model can achieve better information retrieval performance in both overall, and the labeled samples.

Embedding Visualization

We next demonstrate the improvement achieved by the proposed method against the baseline solution. The t-SNE visualization of the proposed method is shown in Figure 3. And the t-SNE visualization of the baseline is shown in Figure 4. According to Figure 3, the known samples (highlighted in blue and purple) are well separated. The ambiguous samples (highlighted in gray) has a visual margin between labeled samples. On the other hand, according to Figure 4, the baseline solution, will map the two types of real data mix with each other, resulting in worse mAP of labeled data (the blue dots and the purple dots are overlapped with each other). The result demonstrates that the proposed method can successfully separate the similar ambiguous samples compared with the labeled sample while preserving the performance in retrieving the labeled sample.

Conclusion

In this paper, we proposed a novel training loss for time series streaming hashing, which widely existed in many time series data related applications. The proposed loss especially considering separating the semantically ambiguous samples, the sample that the semantical meaning is ambiguous but highly similar to the labeled samples, to enhance the robustness of the deep hashing model. The experiment demonstrates that the model trained through the proposed loss can have better performance in both, retrieving the labeled data and distinguish ambiguous samples and the labeled data.

Acknowledgment

This work is supported by the NSF under Grant CNS-2318682 and IIS-2348480. The computational resource is supported by Google Cloud Platform (GCP) credit.

References

Camerra, A.; Palpanas, T.; Shieh, J.; and Keogh, E. 2010. isax 2.0: Indexing and mining one billion time series. In *2010 IEEE International Conference on Data Mining*, 58–67. IEEE.

Cao, Z.; Long, M.; Wang, J.; and Yu, P. S. 2017. Hashnet: Deep learning to hash by continuation. In *Proceedings of the IEEE international conference on computer vision*, 5608–5617.

Chuah, M. C.; and Fu, F. 2007. ECG anomaly detection via time series analysis. In *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops: ISPA 2007 International Workshops SSDSN, UPWN, WISH, SGC, ParDMCom, HiPCoMB, and IST-AWSN Niagara Falls, Canada, August 28-September 1, 2007 Proceedings 5*, 123–135. Springer.

Dau, H. A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C. M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C. A.; and Keogh, E. 2019. The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6): 1293–1305.

Egarter, D.; and Elmenreich, W. 2015. Autonomous load disaggregation approach based on active power measurements. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, 293–298. IEEE.

Gao, Y.; and Lin, J. 2017. Efficient discovery of time series motifs with large length range in million scale time series. In *2017 IEEE International Conference on Data Mining (ICDM)*, 1213–1222. IEEE.

Gao, Y.; and Lin, J. 2019. HIME: discovering variable-length motifs in large-scale time series. *Knowledge and Information Systems*, 61: 513–542.

Hewage, P.; Behera, A.; Trovati, M.; Pereira, E.; Ghahremani, M.; Palmieri, F.; and Liu, Y. 2020. Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, 24: 16453–16482.

Karevan, Z.; and Suykens, J. A. 2020. Transductive LSTM for time-series prediction: An application to weather forecasting. *Neural Networks*, 125: 1–9.

Keogh, E.; Lin, J.; and Fu, A. 2005. Hot sax: Efficiently finding the most unusual time series subsequence. In *Fifth IEEE International Conference on Data Mining (ICDM'05)*, 8–pp. Ieee.

Kim, K.-j. 2003. Financial time series forecasting using support vector machines. *Neurocomputing*, 55(1-2): 307–319.

Li, Y.; Leong, H. U.; Yiu, M. L.; and Gong, Z. 2015. Quick-motif: An efficient and scalable framework for exact motif discovery. In *2015 IEEE 31st International Conference on Data Engineering*, 579–590. IEEE.

Mercer, R.; and Keogh, E. 2024. Novelets: A new primitive that allows online detection of emerging behaviors in time series. *Knowledge and Information Systems*, 66(1): 59–87.

Müller, F. L.; Szabó, J.; Sundström, O.; and Lygeros, J. 2017. Aggregation and disaggregation of energetic flexibility from distributed energy resources. *IEEE Transactions on Smart Grid*, 10(2): 1205–1214.

Palpanas, T. 2020. Evolution of a Data Series Index: The iSAX Family of Data Series Indexes: iSAX, iSAX2. 0, iSAX2+, ADS, ADS+, ADS-Full, ParIS, ParIS+, MESSI, DPiSAX, ULISSE, Coconut-Trie/Tree, Coconut-LSM. In *Information Search, Integration, and Personalization: 13th International Workshop, ISIP 2019, Heraklion, Greece, May 9–10, 2019, Revised Selected Papers 13*, 68–83. Springer.

Pyakillya, B.; Kazachenko, N.; and Mikhailovsky, N. 2017. Deep learning for ECG classification. In *Journal of physics: conference series*, volume 913, 012004. IOP Publishing.

Sezer, O. B.; Gudelek, M. U.; and Ozbayoglu, A. M. 2020. Financial time series forecasting with deep learning: A systematic literature review: 2005–2019. *Applied soft computing*, 90: 106181.

Shokoohi-Yekta, M.; Chen, Y.; Campana, B.; Hu, B.; Zakaria, J.; and Keogh, E. 2015. Discovery of meaningful rules in time series. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, 1085–1094.

Song, D.; Xia, N.; Cheng, W.; Chen, H.; and Tao, D. 2018. Deep r-th root of rank supervised joint binary embedding for multivariate time series retrieval. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2229–2238.

Su, S.; Zhang, C.; Han, K.; and Tian, Y. 2018. Greedy hash: Towards fast optimization for accurate hash coding in cnn. *Advances in neural information processing systems*, 31.

Tsay, R. S. 2005. *Analysis of financial time series*. John wiley & sons.

Wang, Q.; and Palpanas, T. 2023. SEAnet: A Deep Learning Architecture for Data Series Similarity Search. *IEEE Transactions on Knowledge and Data Engineering*.

Yagoubi, D. E.; Akbarinia, R.; Masseglia, F.; and Palpanas, T. 2017. Dpisax: Massively distributed partitioned isax. In *2017 IEEE International Conference on Data Mining (ICDM)*, 1135–1140. IEEE.

Zhu, H.; Long, M.; Wang, J.; and Cao, Y. 2016. Deep hashing network for efficient similarity retrieval. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 30.