Information Sciences 642 (2023) 119147

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

INSTINCT: Inception-based Symbolic Time Intervals series classification

Omer David Harel [a,*], Robert Moskovitch [a,b]

[a] Software and Information Systems Engineering, Ben Gurion University of the Negev, Be'er Sheva, 8410501, Israel
[b] Population Health Science and Policy, Icahn School of Medicine at Mount Sinai, New York, 10029, NY, USA

ARTICLE INFO

Dataset link: https://github.com/omerh18/INSTINCT

Keywords:
Symbolic Time Intervals
Classification
Deep learning
Inception

ABSTRACT

Symbolic Time Intervals (STIs) describe events having a non-zero time duration, which occur in a wide range of application domains. In this paper, we target the challenge of STIs series classification (STIC), which refers to the categorization of series of STIs. Over the recent years several advancements have been made in STIC, all of which are based on either distance-metrics or feature-based traditional classifiers, mostly relying on hand-engineering of features. Due to the high computational cost of either distance calculation or feature extraction, most methods also have quite little potential to scale. We introduce INSTINCT – a novel deep learning-based framework for STIC, which 1) proposes an almost fully information-preserving transformation of raw STIs series into real matrices, and 2) presents a novel ensemble of deep inception-based convolutional neural networks for their classification. The evaluation is applied to the six real-world STIC benchmark datasets and demonstrates that INSTINCT *significantly improves accuracy* over seven state-of-the-art methods, as well as over three deep learning-based baselines. In addition, a comprehensive architecture study of INSTINCT is conducted as well as a scalability analysis, reporting an overall time complexity which is *linear* in each of the main properties of the input STIs series.

1. Introduction

Symbolic Time Intervals (STIs) series describe sets of events that occur over time, in which each event may have a non-zero time duration. For example, the period of time the green light is on in a traffic light or the time period a patient is prescribed on a medication. Hence, compared to the use of instantaneous events only, temporal data representation as series of STIs offers a richer, more flexible representation, which also provides further insights into the underlying events and the temporal relations among them [36].

Over the recent years, a constant growth has been observed not only in the availability of temporal data, but also in the data heterogeneity (e.g., diverse variables, sampling frequencies, time durations etc.). Thus, STIs series data as well as algorithms processing them, have recently become highly valuable in a large variety of real-world application domains. That is, including the classification or outcome prediction in Electronic Health Records (HER) [3,28,29], sign language transcriptions [34], and human activity recognition [22]. However, alongside the promising potential, the high data volumes also impose a burden on such algorithms, which ought to scale well to properly address real-world scenarios. In that respect, deep learning-based methods, which are capable of

* Corresponding author.
E-mail address: omerdavi@post.bgu.ac.il (O.D. Harel).

https://doi.org/10.1016/j.ins.2023.119147
Received 23 January 2023; Received in revised form 7 May 2023; Accepted 10 May 2023

Available online 16 May 2023

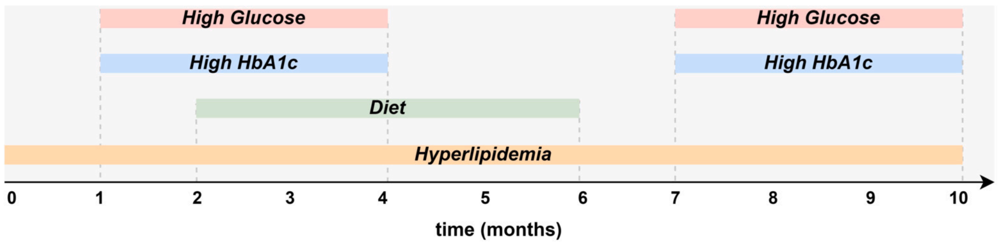


Fig. 1. Illustration of a series of STIs taken from the healthcare domain, representing a clinical record of a patient potentially suffering from diabetes type 2, collected over a time period of ten months.

straight-forwardly utilizing GPU parallel computation abilities, may be of particularly high interest to the research community as well as to relevant industrial use cases.

Fig. 1 illustrates an example of a series of STIs taken from the healthcare domain, which represents a clinical record of a patient potentially suffering from diabetes type 2, collected over a time period of ten months. At first, the patient was diagnosed with hyperlipidemia, i.e., an excess of lipids or fat in their body. Then, high levels of glucose as well as of HbA1c (i.e., percentage of red blood cells having sugar-coated hemoglobin) had been measured, due to which the patient sought medical advice, in which he was recommended a diet. The patient followed their diet for four months. During this time period, glucose and HbA1c levels decreased back to normal, while fat levels were still high. Eventually, after giving-up diet, levels of both glucose and HbA1c increased out of the normal range of values again.

In this paper, we focus on the problem of STIs series classification (STIC), which refers to the categorization of entities that are described by series of STIs. In Fig. 1, the problem of STIC may, for example, refer to the automatic categorization of patients into diabetic or non-diabetic based on their collected measurements. Alternatively, it may refer to the prediction whether or not a patient will be diagnosed with diabetes type 2 during a specified time period (e.g., during the next half-a-year). Thus, potentially assisting in early diagnosis and providing a suitable drug treatment, if necessary.

Since STIs have arbitrary-long time durations, they may overlap and form more complex temporal relations compared to other forms of temporal data (e.g., time series or sequential data), in which events are instantaneous and may be orderly represented. These temporal complexities should therefore be well-captured and modeled by algorithms for STIC. In that respect, the problem of STIC somewhat resembles computer vision tasks, in which spatial information within images should be utilized for image classification, object detection etc. In the field of computer vision, modern deep convolutional neural networks (CNNs) have unarguably been the most successful, while only recently attention-based methods were also demonstrated capable of achieving comparable results [23]. Methods based on convolutional kernels and convolutional neural networks have, in fact, recently took-off in the related task of time series classification as well [5,14], reaching state-of-the-art performance. Therefore, the idea of using advanced convolution-based neural architectures for STIC, combined with a transformation of the raw STIs series data into an appropriate representation to which they can be effectively applied, seems to hold a promising potential for improvement.

Inception-based CNNs, in particular, have a significant potential advantage over conventional CNNs in the context of STIC. The inception module was primarily proposed in [40] for image recognition, and since then it has been further improved and extended in [12,39,41]. While the inception module serves as a convolutional layer as part of a CNN, it enables to apply multiple convolutional kernels of varied sizes in parallel to the input images. That is unlike a conventional convolutional layer, in which only a uniform kernel size is used. When applied to temporal data, this property of the inception module may potentially enable the detection of useful patterns of varied lengths within the data. As will be demonstrated in Section 5, this is one of the keys for the promising classification results of the framework proposed in the paper, when processing raw STIs series of various time spans and densities.

Existing methods for STIC can be broadly divided into distance-based or feature-based classifiers. Distance-based classifiers, among which are Artemis [16] and IBSM [17], define a measure to quantify the distance between two STIs series. The computed distances are then typically used in a k-NN formulation for classification. More recent methods, including KarmaLegoSification [27], STIFE [4], Z-Embedding [20], and SMILE [31]; however, are feature-based. Such methods mostly extract either static, aggregative features or features that are based on variations of temporal patterns or shapelets mined from the STIs series data. The extracted features are then fed into traditional off-the-shelf classifiers (e.g., SVM, BDT), whose classification results are, of course, dependent on the quality of the pre-computed features, and their degree of expressiveness of the classification-informative aspects within the underlying data. That is unlike a deep learning approach as we present in this paper, which aims to automatically learn the feature representation directly from the data, and as part of the classification task.

In addition, several existing methods also rely on predefined parameters, e.g., support and gap [20], that sometimes require domain-specific knowledge and should be set differently for each dataset, which limits their flexibility to some extent. As described in [31], due to the high computational cost of either distance calculation or feature extraction, most of these methods also have little potential scaling to large datasets.

The main contributions of the paper are the following.

- **Novelty.** We introduce INSTINCT – a novel deep learning-based framework for STIC, including:
  - An almost fully information-preserving transformation of raw STIs series into real matrices, which capture the series' STIs that are active over time.

    - Classification of the transformed matrices via a novel ensemble of deep inception-based convolutional neural networks for STIC, inspired by the Inception-v4 network for image recognition [39].
- **Evaluation.** The paper presents a rigorous and extensive evaluation of the proposed framework which is threefold.
    - **Classification performance.** INSTINCT is compared to seven state-of-the-art methods, as well as to three deep learning-based baselines applied to the transformed representation, when evaluated on the six real-world single-label STIC benchmark datasets [25,30]; demonstrating superior performance in terms of both the accuracy, where statistical significance is shown, and AUC.
    - **Scalability.** An empirical scalability analysis of INSTINCT is presented, reporting a linear dependency between its overall runtime and each of the main properties of the input STIs series datasets.
    - **Architecture study.** A comprehensive study of INSTINCT's network architecture is conducted, focusing on the trade-off between the classification accuracy and training time, as well as on the convergence and sensitivity of the proposed framework.
- **Code and data availability.** The source code of INSTINCT, as well as the code and datasets used for the experiments, have been made publicly available.[1]

The rest of the paper is organized as follows: Section 2 formalizes the problem definition of STIC and reviews the related work in this area of research. Section 3 introduces the proposed INSTINCT framework, elaborating on its core building blocks, and Section 4 details the experimental setup designed for the framework's evaluation. Finally, Section 5 reports the experimental results and Section 6 concludes the paper and discusses future research directions.

2. Background

In this section we first provide a few preliminary definitions, followed by the STIC problem formulation. Then, we present the evolvement of the field of STIC along the past two decades, reviewing the major strengths and weaknesses of state-of-the-art methods.

2.1. Problem setting: Symbolic Time Intervals series classification

Definition 1 *(STI).* Let Σ be an alphabet of symbols (i.e., event-types). A *Symbolic Time Interval (STI)* $I = (symbol, s, f)$ is a triplet of a symbol, a start-time, and a finish-time, such that $I.symbol \in \Sigma$ and $I.s < I.f$. An STI I is considered *active* during its time span, i.e., from $I.s$ to $I.f$.

Definition 2 *(STIs Series).* An *STIs series* $S = (I_1, I_2, \dots, I_k)$ is a lexicographically sorted series of STIs, such that $\forall I_i, I_j : i < j \equiv I_i.s < I_j.s \vee (I_i.s = I_j.s \wedge I_i.f < I_j.f) \vee (I_i.s = I_j.s \wedge I_i.f = I_j.f \wedge I_i.symbol < I_j.symbol)$, i.e., S is first sorted by the STIs' start-time, then by their finish-time, and finally by their symbol. The *length* of S refers to its total time span, i.e., $|S| = \max_{1 \le j \le k} I_j.f - I_1.s$, while its *size* is defined by the number of STIs within S, i.e., k.

The STIC problem formulation, which we tackle in this paper, is then defined as follows.

Definition 3 *(The STIC Problem Definition).* Let a collection of STIs series $X = \{S_1, \dots, S_n\}$, a set of predefined class labels Y, and a dataset $D = \{(S_1, y_1), \dots, (S_n, y_n)\}$, in which $\forall i : S_i \in X$ is an STIs series, and $y_i \in Y$ is its corresponding class label. The *STIC problem* is to learn a classifier f on D, which maps any (possibly unseen) input STIs series S to a probability distribution over the set of class labels Y.

Example 1. Fig. 2 illustrates a labeled STIC dataset $D = \{(S_1, y_1), (S_2, y_2), (S_3, y_3)\}$, in which $\forall 1 \le i \le 3$: S_i is an STIs series defined over the symbols alphabet $\Sigma = \{A, B, C\}$, and $y_i \in \{0, 1\}$ is its corresponding class label, which stands for a binary classification task. In S_1, for example, there are two STIs having the symbol A (green), two additional STIs having the symbol B (orange), and three STIs having the symbol C (blue). The earliest STI in S_1 which has the symbol A is said to be *active* during the time range $[0, 3]$, while the next one is active during $[6, 9]$, and so on. Overall, S_1 includes a total number of seven STIs which span from timestamp 0 to 10, similar to S_3. Therefore, the *size* of both S_1 and S_3 is seven, while their *length* equals to ten time-units. That is unlike S_2, which is eight time-units long and includes a total number of only four STIs. Thus, in the given example, both the series S_1 and S_3 have been assigned the same class-label of 0, as they share much resemblance in terms of their size, length, time durations etc., while S_2 has been assigned the class-label 1.

2.2. Related work

Research in the field of STIC has significantly evolved along the past two decades. State-of-the-art methods for STIC can be broadly categorized into the following main two approaches: distance-based classifiers [16,17], on which earlier studies have focused; and feature-based classifiers [3,4,20,27,30,31], that have been spread more widely in recent times. The majority of state-of-the-art

[1] https://github.com/omerh18/INSTINCT.

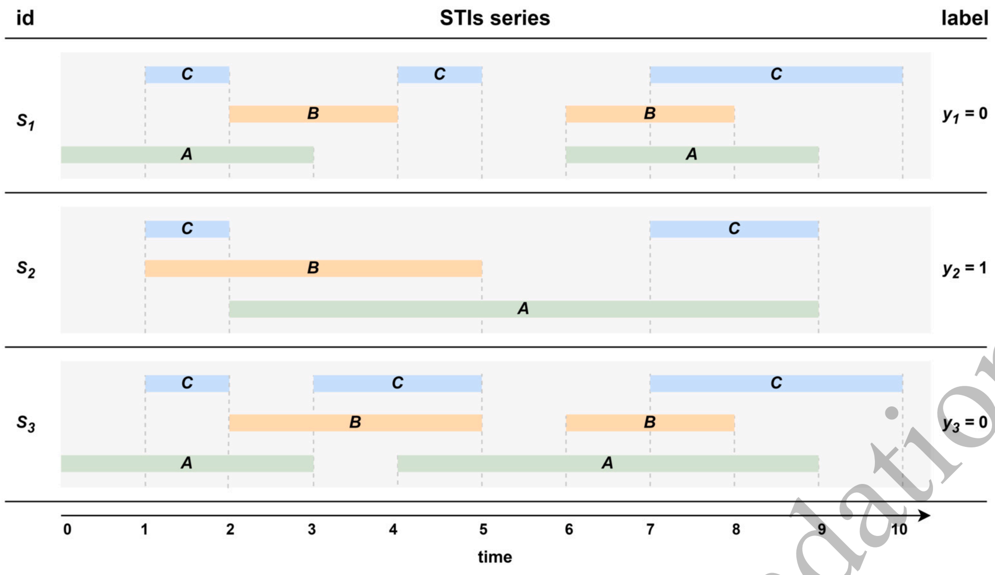


Fig. 2. Illustration of a labeled STIC dataset which includes three STIs series defined over the symbols alphabet $\Sigma = \{A, B, C\}$. The series span along at most 10 time-units, and they include a varying number of STIs – from 4 in S_2 to 7 in both S_1 and S_3. Class labels $y_{1\leq i\leq 3} \in \{0,1\}$, which stands for a binary classification task.

methods for STIC – either distance-based or feature-based, utilize the temporal relations that hold between pairs of STIs, which are most commonly defined using Allen's finite set of 13 temporal relations [1]. Allen's relations include *before, meets, overlaps, starts, contains, finished-by*; their corresponding inverse relations: *after, met-by, overlapped-by, started-by, during, finishes*; and *equals*, which is its own inverse relation.

2.2.1. *Distance-based methods*

Distance-based methods for STIC mostly define a measure to quantify the distance between two STIs series. The computed distances are then typically used in a k-NN formulation for classification.

Artemis [16] quantifies the distance between two STIs series based on the fraction of pairwise temporal relations (i.e., temporal relations between pairs of STIs) that they have in common. The higher the fraction of pairwise temporal relations that the two series share, the smaller the distance between them. Yet, the actual time durations of neither the underlying STIs nor the pairwise temporal relations between them are utilized for classification, which potentially limits the method's predictive power. As elaborated in [20], this method is also quite expensive computation-wise, having a time complexity of $O(|S_1| \cdot |S_2| \cdot |\Sigma|)$ for calculating the distance between each pair of STIs series S_1 and S_2 defined over the symbols alphabet Σ.

In [17], IBSM was proposed as a more informative distance measure. IBSM represents each time point along the entire STIs series' time span as a binary vector, which indicates for each symbol whether or not there is an STI having that symbol which is active during the said time point. To calculate the distance between two STIs series, vectors are then re-sized into a uniform (maximal) size through interpolation. Thus, in contrast to Artemis, IBSM does consider the time durations of the STIs, but does not explicitly refer to the temporal relations among them for distance measuring. In terms of computational complexity, while distance computation is relatively fast, resizing might yield substantially longer runtimes. That is especially in datasets showing high variance in the STIs series' lengths as well as in the number of symbol types that they include. Overall, calculating the distance between two STIs series S_1 and S_2 defined over the symbols alphabet Σ in IBSM, yields a time complexity of $O(|\Sigma| \cdot |S_1| + |\Sigma| \cdot |S_2| + |\Sigma| \cdot \max_{S_i \in D} |S_i|)$, which sums-up to $O(|\Sigma| \cdot \max_{S_i \in D} |S_i|)$.

2.2.2. *Feature-based methods*

Feature-based methods mostly extract either static, aggregative features or features that are based on variations of temporal patterns (shapelets) mined from STIs data or even sequential data. The extracted features are then fed into common classifiers (e.g., SVM, BDT), whose classification results are, of course, directly dependent on the degree to which the preconceived features represent all the classification-informative aspects within the underlying STIs data.

Early studies in the field of feature-based STIC [3,27,30] were mostly applied to healthcare temporal measurements. These measurements have not been always represented as STIs in their raw form, but rather as multivariate time series, which are first abstracted into STIs series representation via temporal abstraction [26,27]. Then, variations of temporal patterns or shapelets of any size are typically extracted. For that, efficient mining algorithms of either sequential patterns [2], association rules [32], or TIRPs, i.e., time intervals-related patterns [9,28] are taken advantage of. Finally, the discovered shapelets are used as features for classification. However, despite the rich information of the discovered arbitrary-long shapelets, the predictive power of most of these methods might still be limited due to the inexpressiveness of the shapelets' time durations in the classification features. In addition, due to the exponential time complexity of the preliminary process of pattern mining [9,27], most of these methods also have a quite little potential to scale.

More recently, in [4], the STIFE framework has been proposed. STIFE extracts both 1) static, aggregative features; 2) class-medoid features, using the IBSM distance measure [17]; and 3) class-distinctive pairwise temporal relations from the STIs series, which are then fed into a random forest classifier. Due to the large variety of extracted features, STIFE achieved significantly improved classification performance. Yet, the slight information exploited regarding the STIs' time durations, alongside the inability to capture potentially valuable TIRPs that include more than just a pair of STIs, still limit the framework's predictive power. In addition, as described in [4], STIFE's memory requirements might be impractical for large symbols alphabet sizes $|\Sigma|$. Assuming a dataset of n STIs series of size m, STIFE's training time and memory have quadratic complexity in both n, m, and $|\Sigma|$. Given a single STIs series, its feature extraction time complexity is $O(m \cdot (log(m) + |\Sigma|))$, which might be longer compared to IBSM.

In [20], a novel spectral embedding of STIs series was proposed, called Z-Embedding. The embedding is based on the pairwise temporal relations that the STIs series include and is later used in several traditional classifiers for classification. First, the entire STIs series dataset is converted into a bipartite graph representation with two vertex sets of (1) STIs series and (2) pairwise temporal relations. Then, a pruning step is performed based on predefined minimal and maximal support (frequency) thresholds, as well as a maximal gap constraint, which limits the allowed time duration between two STIs among which the temporal relation is *before*. Note that the appropriate values for these parameters may differ from one dataset to another and setting them might sometimes also require domain-specific knowledge, which makes the method less plug-and-playable. Finally, the graph is represented as a bi-adjacency matrix from which feature vectors are created through regularization and singular value decomposition (SVD). Similar to STIFE, this method does not exploit information regarding neither the STIs' time durations nor potentially valuable TIRPs that include more than two STIs. Assuming a dataset of n STIs series of size m, the time complexity of creating the bi-adjacency matrix is $O(n \cdot m^2 + n \cdot |\Sigma|^2)$, while the process of SVD has a time complexity of $O(n \cdot |\Sigma|^2 \cdot min(n, |\Sigma|^2))$.

Building upon STIFE, the SMILE framework was introduced in [31], achieving state-of-the-art classification accuracy in STIC. Beyond the extraction of static features, class-medoid features, and pairwise temporal relations employed in STIFE; SMILE also extracts features from *e-lets*, which are random sub-series of potentially more than just a pair of STIs. These additional features aim to address the main deficiencies of STIFE, by capturing potentially valuable information found in either the STIs' time durations or in shapelets that include an arbitrary large number of STIs. However, as reported in [31], SMILE's feature extraction has a high computational cost, having its overall time and memory complexity inferior to STIFE.

In the next section we introduce the INSTINCT framework for STIC. INSTINCT is inspired, inter alia, by recent advancements made in convolutional neural networks and in deep learning-based temporal data processing. Hence, before elaborating on INSTINCT, we first present succinct reviews of the related research fields of computer vision and time series classification, where advanced deep learning architectures in general, and convolutional neural networks in particular, have widely spread in recent times.

2.2.3. *Computer vision*

The use of deep learning in the field of computer vision has significantly evolved over the recent decades. Specifically, as stated in [23], convolutional neural networks (CNNs) have been the de-facto standard for a large variety of computer vision tasks for the past few years. Starting with LeNet-5 [19], CNNs have typically had a common structure. That is, using several stacked convolutional layers (possibly combined with max pooling) followed by at least a single fully-connected layer. However, the use of CNNs for computer vision tasks has really took-off since AlexNet [18] achieved the best classification results, at that time, on ImageNet. Since then, the quality of backbone CNN architectures significantly improved either by going deeper or wider. Such backbone networks include VGGNet [38], ResNet [10], and GoogLeNet [40], which is also referred as Inception-v1.

The original inception paper [40] presented a CNN architecture for image recognition relying on the inception module. While the inception module serves as a convolutional layer as part of a CNN, it enables to apply multiple convolutional kernels of varied sizes in parallel to the input images. That is unlike a conventional convolutional layer, in which only a uniform kernel size is used. Thus, potentially increasing the flexibility of the network and the ability to detect useful patterns of diverse sizes within visual inputs. Since then, the inception network has been further extended forming Inception-v2 with the introduction of batch normalization [12], Inception-v3 using additional factorizations [41], and Inception-v4 and Inception-resnet [39], adding residual connections.

However, deep learning-based methods for computer vision are, of course, not limited to CNNs. Following the resounding success of attention and transformers in the field of natural language processing, they have recently propagated into the vision community, achieving at least comparable results to CNN-based architectures in tasks such as object detection and image segmentation [23]. In addition, as described in [46], the use of multimodal features (e.g., language information, user clicks, etc.) beyond just visual features, was found highly beneficial for image retrieval [45] as well as for fine-grained image classification [46,48]. Hand-crafted visual feature descriptors were found useful as well for image feature representation [47], especially when seeking for better interpretability.

We highlight that this subsection by no means intends to present a complete review of the field of computer vision, but rather a concise review, which lays the relevant foundations for the introduction of the proposed INSTINCT framework for STIC in Section 3.

2.2.4. *Time series classification*

Until recently, time series classification methods have mostly relied on traditional off-the-shelf classifiers, employed on a preconceived feature representation of the raw data. Based on the nature of the extracted features, these methods can be broadly categorized into distance-based [37], dictionary-based [33], and shapelets-based [11] methods, as well as transformation ensembles [21]. Following similar principles, several enhanced methods have been recently proposed as well [24,35], motivated by the relatively high computational complexity of earlier methods.

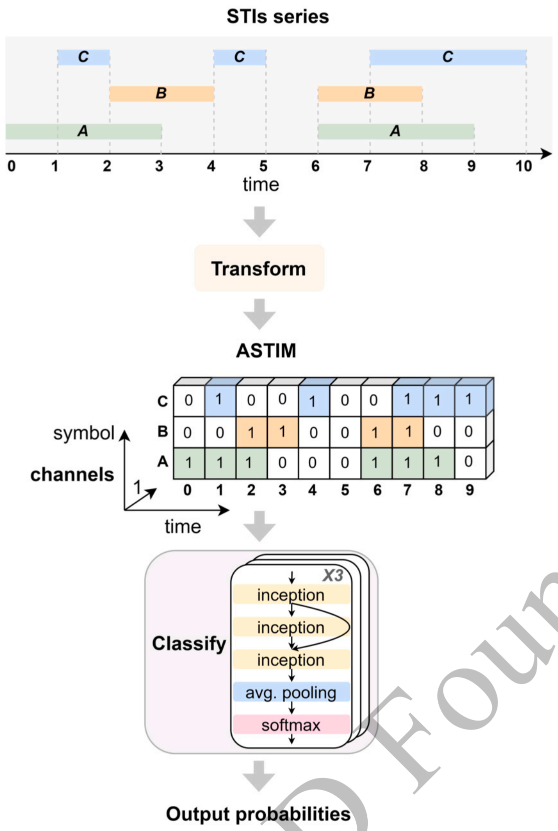


Fig. 3. Illustration of INSTINCT's pipeline – from an input raw STIs series into an output probability distribution over class labels. Initially, the input series is transformed into a two-dimensional representation as a matrix which captures the series' STIs that are active over time. Then, the transformed matrix is fed into INSTINCT's ensemble of three deep CNN models for STIC. Final classification results are obtained through averaging the output probability distributions of the three individual models.

Over the recent years, however, another approach for time series classification has been developed, taking advantage of convolutional kernels and convolutional neural networks [5,14,43]. When applied to raw time series data, convolutions can be seen as sliding one-dimensional filters over raw time series, which unlike images, exhibit only a single time dimension instead of two spatial dimensions [13]. In particular, several inception-based methods have been recently proposed for time series classification, either relying on pre-trained or custom architectures [14]. The most recent advancements in convolution-based time series classification, e.g., InceptionTime [14] and ROCKET [5], have reached state-of-the-art accuracy while also reporting significant scalability. Since then, several enhanced variations of ROCKET have been proposed as well, aiming to further improve either runtime [6] or accuracy [42].

Finally, as described in [13], Recurrent Neural Networks (RNNs) have rarely been applied for time series classification, but rather for the purpose of time series forecasting. More advanced concepts arising from the field of natural language processing, such as attention and transformers, have made their very first steps in time series classification only recently in [44,49].

3. INSTINCT: Inception-based Symbolic Time Intervals series classification

In this section we introduce INSTINCT – a novel deep learning-based framework for STIC. INSTINCT is comprised of two main phases – Transform and Classify. First, the complete pipeline of INSTINCT is described from a bird's-eye view, after which we elaborate on each of its two main phases.

An overview of INSTINCT's flow – from an input raw STIs series into an output probability distribution over class labels, is illustrated in Fig. 3 (top to bottom). Initially, the raw STIs series at the top is transformed into a two-dimensional representation as a matrix which captures the series' STIs that are active over time. Unlike images, this matrix exhibits only a single spatial dimension, i.e., the time dimension, while the STIs' symbol types (e.g., A, B, and C) are represented as channels, as shown in the middle of Fig. 3. We refer to the transformed matrix as an ASTIM, which we will formally define in the next subsection.

Then, the transformed ASTIM is fed into INSTINCT's novel deep CNN architecture for STIC, which is inspired by the Inception-v4 network for image recognition [39]. In that respect, it is important to highlight that the initial transformation by no means intends to perform feature extraction, but rather to preserve the raw information of the input STIs series through the created ASTIMs to the maximum extent. That is to enable the network to automatically learn an appropriate representation of the underlying data, rather than relying on a preconceived, hand-crafted feature representation. As illustrated at the bottom of Fig. 3, INSTINCT also uses an ensemble of three classifiers, which are identical except for their randomly initialized weight values, to increase the robustness of the proposed framework. The final output probability distribution is then obtained through averaging the probability distributions of each of the three individual classifiers.

3.1. Phase I: Transform – STIs series representation as ASTIMs

INSTINCT's initial Transform phase converts raw input STIs series (Definition 2) into a two-dimensional representation as real matrices, which capture the series' STIs that are active over time. The transformed matrices are referred as *Active-STIs Matrices (ASTIMs)*, which are defined as follows.

Definition 4 *(activity vector)*. Let S be an STIs series defined over the symbols alphabet Σ, and a symbol $\sigma \in \Sigma$. The *activity vector* of σ in S is a $|S|$ dimensional vector a_σ^S such that: $\forall 0 \le t < |S| : a_\sigma^S[t] = |\{I \in S | I.symbol = \sigma \wedge I.s \le t < I.f\}|$. That is, $a_\sigma^S[t]$ stands for the *number* of STIs in S having the symbol σ that are active during the time-unit t.

Definition 5 *(ASTIM)*. Let S be an STIs series defined over the symbols alphabet Σ. The *Active-STIs Matrix (ASTIM)* of S is a matrix $A^S \in \mathbb{R}^{|\Sigma| \times |S|}$ such that: $\forall \sigma \in \Sigma, 0 \le t < |S| : A_{\sigma,t}^S = a_\sigma^S[t]$, where a_σ^S is the activity vector of σ in S (Definition 4).

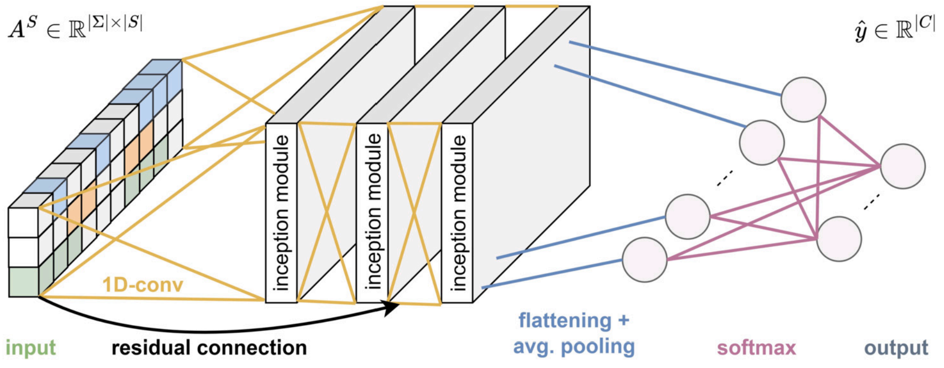
Example 2. At the top of Fig. 3, INSTINCT's transformation of an input raw STIs series S into its corresponding ASTIM A^S (Definition 5) is illustrated. Since S is defined over the symbols alphabet $\Sigma = \{A, B, C\}$, the transformed ASTIM A^S is comprised of three activity vectors – one vector for each symbol type $\sigma \in \Sigma$. Each such vector a_σ^S represents the number of STIs having the symbol σ that are active during each time-unit $0 \le t < |S|$, as defined in Definition 4. For example, at time-unit 0 there is only a single active STI which has the symbol A. Therefore, $a_A^S[0] = 1$, while both $a_B^S[0]$ and $a_C^S[0]$ equal to zero.

Since no ordinal relationship holds between the symbols of an alphabet Σ and their spatial proximity is arbitrary, they are represented as channels within the transformed ASTIMs rather than as another spatial dimension (Fig. 3). This property enables INSTINCT's deep CNN architecture to slide one-dimensional filters over the ASTIMs' time dimension, where each filter applies to all the $|\Sigma|$ channels (i.e., symbol types) together. An additional important benefit of the proposed representation is the ability to explore significantly longer convolutional filters at relatively lower model complexities compared to computer vision tasks. That is due to the one less spatial dimension of the transformed ASTIMs compared to images.

Note that raw STIs series may of course differ in their overall time span, as well as in the number of symbol types that they include. Since the series' ASTIMs are later fed into a deep convolutional neural network for classification, it is mandatory to enforce a uniform ASTIM size amongst all the STIs series in a dataset. For that purpose, in a previous work [17], interpolation has been suggested. In this paper, however, conventional vertical and horizontal zero-padding of ASTIMs (i.e., in both the time and symbol axes) is preferable. That is mainly due to its simplicity and computational efficiency, alongside the fact that the notion of the original length of the STIs series, which has been a key feature for classification in several previous works [4,31], is preserved. Overall, given a dataset of n input STIs series $D = \{S_1, \dots, S_n\}$ defined over the symbols alphabet Σ, the Transform phase can be formulated as a function $f : D \to \mathbb{R}^{|\Sigma| \times \max_{S_i \in D} |S_i|}$ such that $\forall S_i \in D : f(S_i) = A^{S_i}$, for S_i's ASTIM A^{S_i} as defined in Definition 5.

When multiple STIs which have the same symbol do not overlap, in which case the transformed ASTIMs are completely binary, the original STIs series can be fully reconstructed from them (a pseudo-code for raw STIs series reconstruction is provided in Appendix B). This property stands for the invertibility of the Transform function, which indicates that the complete information of raw STIs series is preserved through their ASTIMs, and that no information is lost prior to the classification. Otherwise, when multiple STIs which have the same symbol indeed overlap, the ASTIMs' activity vectors are no longer binary, but rather they count the total number of active STIs in each time-unit, as defined in Definitions 4–5. In such case, we note that only these specific overlapping STIs cannot be fully reconstructed from the transformed ASTIMs, since their start and finish end-points cannot be non-ambiguously determined. However, although theoretically possible, in practice, this scenario is quite rare, as in many application domains it does not always make sense to have several events of the same type occurring simultaneously. In fact, amongst all the real-world STIC benchmark datasets [25,30], such non-binary ASTIM cells constitute even less than 0.3% of the cells of all the transformed ASTIMs. Thus, although it might be an interesting theoretical challenge, the amount of raw information hidden from the classifier building upon the proposed representation, seems to be practically negligible and is expected to have only minimal impacts on classification.

Complexity. Let $D = \{S_1, \dots, S_n\}$ be a dataset of n m-sized STIs series defined over the symbols alphabet Σ. A naïve implementation of INSTINCT's Transform phase may traverse the series' STIs in an arbitrary order, and incrementally increase the counts of their activity time frames within the transformed ASTIMs. Thus, yielding a time complexity of $O(m)$ for a single STIs series, and $O(n \cdot m)$ for the complete dataset, if no parallelism is utilized (although indeed possible). Memory complexity for the entire dataset sums to $O(n \cdot |\Sigma| \cdot \max_{S_i \in D} |S_i|)$, which is the total size of all the STIs series' ASTIMs. Note that since the ASTIMs' dimensionality depends on the length of the series, this representation might be somewhat wasteful when processing rather sparse STIs series data. In a future work, we intend to propose an alternative sparse ASTIM representation.



$A^S \in \mathbb{R}^{|\Sigma|\times|S|}$
$\hat{y} \in \mathbb{R}^{|C|}$
inception module
inception module
inception module
1D-conv
input
residual connection
flattening + avg. pooling
softmax
output

Fig. 4. Illustration of INSTINCT's inception-based network architecture for STIC.

3.2. Phase II: Classify – Inception-based classification of ASTIMs

In INSTINCT's Classify phase, the ASTIMs which have been created in the preliminary Transform phase are classified by a novel ensemble of three deep CNN models for STIC, inspired by the Inception-v4 network for image recognition [39]. First, we concisely describe how convolutional layers and convolutional neural networks can be utilized for STIC as part of INSTINCT's pipeline. Then, the proposed inception-based network architecture is presented, followed by a detailed description and formulation of INSTINCT's inception module designed for STIC.

As a convolutional neural architecture, INSTINCT's network relies on convolutional layers, which slide filters over the transformed ASTIMs to extract non-linear time-invariant features from them. Through stacking multiple such layers, higher order, hierarchical features can be extracted as well, which are expected to further improve the classification performance. As described in the previous subsection, ASTIMs exhibit only a single spatial dimension, i.e., the time dimension, whereas the symbols are represented as channels. Therefore, a convolutional layer in INSTINCT slides one-dimensional filters over the ASTIMs' time dimension, where each filter applies to all the symbols together. Due to the one-less spatial dimension compared to images, these filters can be much longer compared to the filters typically employed in computer vision architectures.

INSTINCT's network architecture for STIC, which maps an input ASTIM $A^S \in \mathbb{R}^{|\Sigma|\times|S|}$ into an output probability distribution over class labels $\hat{y} \in C$, is depicted in Fig. 4. First, three inception modules are stacked, such that the output of each module is directly fed as input into the successive module. As will be elaborated later in this subsection, INSTINCT's inception module simultaneously applies multiple one-dimensional convolutional kernels of varied sizes to the input ASTIM, which are then concatenated to form the module's output. In addition, as illustrated in Fig. 4, residual connections are used, as suggested in [39]. That is, to mitigate the problem of vanishing gradients [10]. Through the residual connections, for example, the input of the first inception module in INSTINCT is also added via a shortcut linear connection to the third module's input, which stands for a residual block size of two modules, in similar to ResNet [10].

Next, to convert the output of the third inception module into a feature vector, either flattening and local average pooling (FAP), or global average pooling (GAP) are typically employed. When applied to STIs series data represented as ASTIMs, GAP shrinks the whole arbitrary-long time dimension of the ASTIMs into only a single value. Thus, drastically reducing the number of parameters of the model, and therefore its complexity.

When processing thousands of time-units long STIs series, which indeed occur among the benchmark datasets, this is a quite coarse operation. Hence, in such cases, GAP might not always represent well-enough local classification-informative aspects of the underlying STIs data, and in turn deteriorate predictive performance, as well as result in a somewhat slower, more spiky convergence. Local average pooling, on the other hand, only moderately reduces the input STIs series' length by aggregating over a small sliding window of the ASTIMs, aiming to reach a reasonable balance between locality and globality. Hence, in INSTINCT, employing FAP was preferred over a GAP layer. That is, with great caution to the increased risk of overfitting small datasets, due to the increased complexity of the model.

To yet reduce our model complexity, and as a consequence its likelihood to overfit, we do not employ any additional fully connected layer prior to the final softmax layer, whose number of neurons is equal to the number of target classes in the dataset $|C|$. In addition, we propose a much more compact network architecture compared to typical networks for computer vision, by stacking only three inception modules which are also comprised of much fewer filters, as will be explained shortly. Finally, an ensemble of three classifiers is used, which are identical except for their randomly initialized weight values, to increase the robustness of the proposed framework. The output probability distribution of INSTINCT is then obtained through averaging the probability distributions of each of the three individual classifiers. For completeness, categorical cross entropy loss was used as the loss function of each of the individual classifiers, i.e., $L_{CE} = -\Sigma_{i=1}^{|C|} y_i \cdot log\, \hat{y}_i$. That is since the general case with which the proposed framework deals is not necessarily binary, but rather a multiclass classification problem.

The design of INSTINCT's inception module is depicted in Fig. 5. Recall that the inception module's input is an STIs series S represented as an ASTIM A^S (Definition 5), which has a single $|S|$ time-units long time dimension and $|\Sigma|$ channels, given an alphabet of symbols Σ. Initially, for the purpose of dimensionality reduction, the input ASTIM flows through a bottleneck layer. The

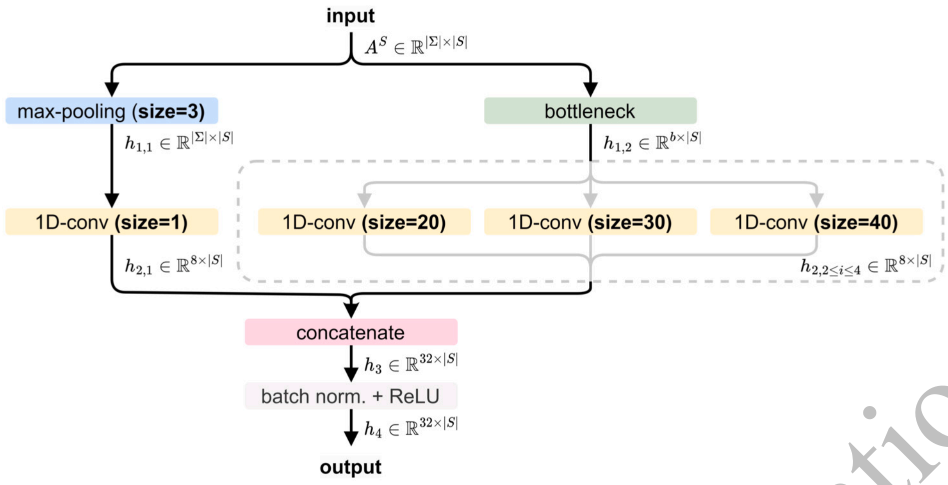

input
$A^S \in \mathbb{R}^{|\Sigma|\times|S|}$
max-pooling (size=3)
bottleneck
$h_{1,1} \in \mathbb{R}^{|\Sigma|\times|S|}$
$h_{1,2} \in \mathbb{R}^{b\times|S|}$
1D-conv (size=1)
1D-conv (size=20)
1D-conv (size=30)
1D-conv (size=40)
$h_{2,1} \in \mathbb{R}^{8\times|S|}$
$h_{2,2\le i\le 4} \in \mathbb{R}^{8\times|S|}$
concatenate
$h_3 \in \mathbb{R}^{32\times|S|}$
batch norm. + ReLU
$h_4 \in \mathbb{R}^{32\times|S|}$
output

Fig. 5. INSTINCT's inception module. First, the dimensionality of the module's input, which is an STIs series represented as an ASTIM, is reduced through the initial bottleneck layer, followed by which one-dimensional convolutional filters of multiple lengths are applied in parallel. The individual convolution outputs, as well as the output of a max pooling layer combined with an additional convolution of length 1, are then concatenated and fed into a batch normalization layer and a ReLU activation, through which the module's output is obtained.

bottleneck layer reduces the channels dimensionality of the ASTIM by applying $b << |\Sigma|$ one time-unit long filters, with a stride of 1. Thus, preserving the original length of the STIs series while significantly reducing the model complexity, which is expected to assist in preventing the model from overfitting.

Following the said bottleneck layer, one-dimensional filters of multiple lengths are simultaneously applied to the ASTIM, as shown on the right-hand side of Fig. 5. This is the core property of the inception module and the key for its high flexibility, which enables the automatic extraction of valuable features and patterns of variable lengths from the STIs series. INSTINCT's inception module uses three sets of parallel convolutional filters which are 20, 30, and 40 time-units long, including exactly 8 filters each. In addition, as shown on the left-hand side of Fig. 5, a parallel max pooling layer is employed as well, whose output is directly fed into another one-dimensional convolutional layer with filters of length 1. That is, to enforce a uniform output dimensionality amongst all the parallel convolutions that are then concatenated. This results in an overall number of $8 \cdot (3+1) = 32$ convolutional filters per module, taking into account the additional 8 filters which follow the max pooling layer. At last, to potentially accelerate convergence, a batch normalization layer is employed, through which the concatenated convolution outputs flow. Then, a ReLU activation is applied, after which the unified output is fed as input into the next layer of INSTINCT's network.

Assume:

$A^S \in \mathbb{R}^{|\Sigma|\times|S|}$ – an input ASTIM
$f_b \in \mathbb{R}^{b\times 1}$ – one-dimensional filters applied in bottleneck layer
$l_1 = 1, l_2 = 20, l_3 = 30, l_4 = 40$ – lengths of applied parallel convolutional filters
$f_{1\le i\le 4} \in \mathbb{R}^{8\times l_i}$ – one-dimensional filters applied in each parallel convolution
MP_3 – max pooling operation with a window size of three time-units
BN – batch normalization operation
$ReLU$ – ReLU activation function, i.e., $ReLU(x) = max(x, 0)$

Overall, INSTINCT's inception module for STIC can be formulated as a function $\phi : \mathbb{R}^{|\Sigma|\times|S|} \rightarrow \mathbb{R}^{32\times|S|}$, such that $\phi(A^S) = ReLU(BN(MP_3(A^S) * f_1 \oplus (A^S * f_b) * f_2 \oplus (A^S * f_b) * f_3 \oplus (A^S * f_b) * f_4))$, as illustrated in Fig. 5.

One final note is that the original length of the input ASTIM remains unchanged throughout the network. When processing datasets with thousands of time-units long STIs series, this property might result in both 1) substantially longer training times, due to sliding the convolutional filters continuously along the whole time dimension; and 2) an increased risk of overfitting, due to a large number of parameters in the final softmax layer, after flattening the ASTIM. Therefore, in such datasets which are also usually rather sparse, we recommend using a stride larger than 1 within each residual block, with a length equals to half the shortest filter of INSTINCT – i.e., $\frac{min\{20,30,40\}}{2} = 10$. Thus, reducing the total number of performed convolutions in the i^{th} residual block of INSTINCT by a factor of 10^i. In addition, the overall length of the ASTIM, and consequently the number of parameters in the final softmax layer, are also reduced by roughly a factor of $10^2 = 100$, having two residual blocks in INSTINCT's network. As described in the previous subsection, in a future work we also propose an alternative approach for processing sparse STIs series datasets, using a sparse ASTIM representation.

The design of INSTINCT's network architecture in terms of both the hyperparameter values (e.g., network depth, ensemble size etc.) and the network layers, and in particular the proposed use of FAP over GAP; has been driven by a comprehensive architecture study which will be presented in experiment 3. The values of the model hyperparameters have been opted on the benchmark datasets, aiming to reach the sweet spot with the most accurate classification results at the lowest model complexity, or training time. The

Table 1
Summary of the main properties of STIC real-world benchmark datasets. Extreme properties appear in bold.

Dataset	n	N	m	l	$\|\Sigma\|$	$\|C\|$
Auslan2	200	2447	12.24	29	12	**10**
Blocks	210	1207	5.75	122	8	**8**
Context	240	19355	**80.65**	283	54	5
Hepatitis	**498**	**53921**	**108.28**	**7554**	63	2
Pioneer	160	8949	55.93	79	**92**	3
Skating	**530**	23202	43.78	**6828**	41	6

selection of FAP over GAP has been empirically substantiated by comparing the two architectures, having the former reporting not only somewhat improved classification performance, but also a typically faster, more stable convergence. A sensitivity analysis of the proposed framework has been conducted as well, increasing the confidence that it does not tend to overfit.

4. Evaluation

4.1. Data

4.1.1. Real-world data

To evaluate the proposed INSTINCT framework, we used the publicly available single-label STIC benchmark datasets [25,30], which include six real-world datasets from various application domains. These datasets, to the best of our knowledge, are all the acknowledged single-label real-world STIC datasets available online, which have been commonly used as well for the evaluation of previous methods [2,4,17,20,31]. Multi-labeled datasets are outside of the scope of this study. Table 1 summarizes the main properties of benchmark datasets considering the following six parameters: (1) number of STIs series in the dataset n (i.e., data samples), (2) total number of STIs N, (3) mean STIs series' size m (i.e., number of STIs within a series), (4) maximal STIs series' length l (i.e., total time span or duration of a series), (5) distinct number of symbol types $|\Sigma|$ (i.e., symbols alphabet size), and (6) the number of class labels $|C|$. The extreme properties of each dataset appear in bold in Table 1. A more detailed description of them is provided in Appendix A.

4.1.2. Synthetic data

As observed in Table 1, the benchmark datasets are relatively small in terms of the number of input data samples that they include n, and they also differ from each other in multiple aspects. Thus, to enable a large-scale analysis of INSTINCT's scalability, in which the effect of each of the input datasets' properties on performance can be isolated, we introduce the following scheme for synthetic STIC datasets generation.

Let a predetermined configuration of a number of STIs series n, a uniform STIs series' size m and length l, and an alphabet of symbols Σ. First, n empty STIs series $\{S_1, \ldots, S_n\}$ of length l are initialized. Then, for each series S_i it is populated with m STIs whose symbol types are drawn uniformly at random from Σ, and their start and finish timestamps are selected uniformly at random from $[0, l]$, such that the start time always precedes the finish time. Class labels are drawn uniformly at random as well from $\{0, 1\}$, forming a completely random binary STIC dataset, which matches the desired configuration of parameter values.

Due to the randomness of class labels, these datasets are of course irrelevant for predictive performance evaluation. However, they are highly useful for a scalability analysis in which the effect of each parameter on the runtime of INSTINCT can be isolated. That is, through running INSTINCT on multiple such datasets in which the value of an examined property is gradually increased, while the values of the rest of the properties are fixed. The experimental setup of the scalability analysis, as well as of the rest of conducted experiments, will be described in detail in the next subsection. For the complete reproducibility of the generated datasets, they are included in our online code and data repository, and the input configurations used for their generation are specified in the experimental setup as well.

4.2. Experimental setup

The experimental setup designed for the evaluation of INSTINCT is threefold. First, INSTINCT's classification performance was compared to state-of-the-art methods for STIC [2,4,17,20,31], as well as to three deep learning-based baselines, on the benchmark datasets. Second, using the described synthetic datasets a scalability analysis has been conducted, in which the potential of the proposed framework to scale to much larger datasets, which is one of the main pitfalls of the majority of state-of-the-art methods, has been evaluated. At last, an extensive architecture study of INSTINCT's network was conducted, aiming to substantiate the design of the proposed network architecture considering predictive performance, convergence, and sensitivity aspects.

All the experiments were run on an HP Omen having 16 GB main memory running Microsoft Windows 10, with a GTX 1660 Ti GPU. Since INSTINCT is the only deep learning-based method for STIC, when testing the complexity of the proposed framework, GPU capabilities have not been utilized and the ensemble classifiers have been run serially. In addition, models were trained using the Adam optimization algorithm [15] for at most 200 epochs, having their weights initialized randomly. Finally, all the comparisons were performed using 10-fold cross-validation. As suggested in [14], the Friedman test [8] had been carried to reject the null

Table 2
Network hyperparameter values' search space.

Hyperparameter	Search space
ensemble size	1–30
residual connections	with/without residual connections
bottleneck layer size	2–32, infinite (i.e., no bottleneck layer)
network depth	1–6
filter length	8–64
number of filters	4–32

hypothesis, followed by a pairwise post-hoc analysis in which the average-rank comparison was replaced by a Wilcoxon signed-rank test with $\alpha = 5\%$ for the comparisons with state-of-the-art methods, and $\alpha = 10\%$ for the architecture study. For visualization, critical difference (CD) diagrams have been used [7], in which non-significantly different methods are connected by a thick horizontal line.

To contribute to future research in the field of STIs series data processing, and allow easy and complete reproducibility of our experimental results, all the following were made publicly available on our online code and data repository:

- Source code of INSTINCT
- Code for all the experiments
- The experimental results in CSV files
- Real-world and synthetic datasets
- Synthetic datasets generator

4.2.1. *Experiment 1: Classification performance*

INSTINCT's classification performance was compared to seven state-of-the-art methods on the STIC benchmark datasets (Table 1). The compared methods include SPAM [2] as a representative of sequential patterns-based methods, 1-nearest neighbor using the IBSM distance metric [17], and the feature-based classifiers: STIFE [4], Z-Embedding [20] and SMILE [31], as well as its two sub-methods MEDOID and STATIC. In addition, to particularly evaluate the predictive power of INSTINCT's network architecture beyond just the use of deep learning, it was also compared to three deep learning-based baselines applied directly to INSTINCT's ASTIMs representation. The evaluated baselines include 1) a feed-forward network, 2) an LSTM network, and 3) a conventional CNN with a uniform filter size of 30, which is the average length of filters applied in INSTINCT's inception module. Since few of the benchmark datasets are imbalanced, evaluation metrics in this experiment have also included the AUC metric, beyond just the methods' accuracy scores.

4.2.2. *Experiment 2: Scalability analysis*

The goal in this experiment was to empirically evaluate INSTINCT's potential of scaling to much larger datasets. Specifically, we analyzed how each of the main four properties of STIC datasets, i.e., the number of STIs series (i.e., training samples), their size and length, and the number of symbol types (i.e., symbols alphabet size); affects the computation time of each of the two main phases of INSTINCT – Transform and Classify. For each examined property, INSTINCT was run on multiple synthetic datasets in which the value of the said property was gradually increased, while the values of the rest of the properties were fixed. Computation times were averaged over 5 repeated runs. The input configurations used for the generation of synthetic datasets are specified in Appendix C.

Based on the analytical complexity analysis of the Transform phase provided in Subsection 3.1, a time complexity which is linear in both the number of STIs series and the series' size was hypothesized, while the STIs series' length and the number of symbol types were expected to have no influence on this phase's runtime. On the other hand, the runtime of the Classify phase, i.e., the network training time, was expected to depend only on the STIs series' length and the number of symbol types, which define the dimensionality of the transformed ASTIMs (Definition 5), as well as on the total number of input data samples.

4.2.3. *Experiment 3: Architecture study*

In this experiment, a comprehensive architecture study of INSTINCT's network had been conducted, by which the design of the proposed network architecture was guided. The architecture study is threefold.

Hyperparameters study. Investigation of the main hyperparameters of INSTINCT's ensemble of deep CNN models as well as of the inception module, among which are: the ensemble size, usage of residual connections, bottleneck layer size, network depth, filter lengths, and the number of filters. In this experiment, the values of the described hyperparameters were opted on the benchmark datasets, focusing on the trade-off between improved classification accuracy and reduced model complexity and training times. Note that to avoid the problem of overfitting, the latter must have been treated with great caution. That is mainly due to the relatively small size of benchmark datasets in terms of the number of STIs series (i.e., training samples) that they include, which practically limits the explorable complexity of the models aiming to mitigate the risk of overfitting. In accordance, the hyperparameter values' search space explored in this experiment is outlined in Table 2.

FAP vs GAP. In this experiment, the proposed usage of flattening and average pooling (FAP) over global average pooling (GAP) in INSTINCT, for the conversion of the inception module's output into a feature vector, was investigated. For that, INSTINCT was run once using FAP as presented in the paper (Fig. 4), and then using a GAP layer instead, on the benchmark datasets. First, the

Table 3
Accuracy scores comparison on benchmark datasets. In each dataset, best performing methods appear in bold, while second best scores are underlined.

Dataset	Accuracy										
	SPAM	STATIC	MEDOID	IBSM	STIFE	SMILE	Z-Embedding	$base_{FF}$	$base_{LSTM}$	$base_{CNN}$	INSTINCT
Auslan2	0.32	0.305	0.46	0.4	0.505	0.485	0.325	0.47	<u>0.57</u>	0.53	**0.63**
Blocks	0.138	0.933	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.976	<u>0.992</u>	0.984	**1.0**
Context	0.888	0.954	0.961	0.967	<u>0.992</u>	0.988	0.958	0.925	0.983	0.892	**1.0**
Hepatitis	0.697	0.663	0.697	0.631	0.799	0.813	<u>0.827</u>	0.715	0.795	0.737	**0.91**
Pioneer	0.773	0.822	0.9	0.956	0.977	0.981	0.981	0.778	<u>0.993</u>	0.847	**0.994**
Skating	0.813	0.674	0.919	<u>0.983</u>	0.966	0.977	0.91	0.945	0.911	0.866	**0.992**
Mean	0.605	0.725	0.823	0.823	0.873	<u>0.874</u>	0.834	0.802	<u>0.874</u>	0.809	**0.921**

Table 4
AUC scores comparison on benchmark datasets. In each dataset, best performing methods appear in bold, while second best scores are underlined.

Dataset	AUC										
	SPAM	STATIC	MEDOID	IBSM	STIFE	SMILE	Z-Embedding	$base_{FF}$	$base_{LSTM}$	$base_{CNN}$	INSTINCT
Auslan2	0.769	0.585	0.748	0.674	0.631	0.686	0.763	0.831	0.833	<u>0.857</u>	**0.88**
Blocks	0.51	0.899	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	0.998	0.998	<u>0.999</u>	**1.0**
Context	0.968	0.994	0.9	0.98	**1.0**	**1.0**	<u>0.998</u>	0.973	0.996	0.95	**1.0**
Hepatitis	0.757	0.695	0.836	0.641	0.854	<u>0.863</u>	0.812	0.693	0.777	0.71	**0.901**
Pioneer	0.734	0.892	0.964	0.97	0.995	0.997	0.997	0.814	<u>0.999</u>	0.835	**1.0**
Skating	0.955	0.914	0.977	0.989	<u>0.997</u>	**0.999**	0.984	0.972	0.981	0.928	**0.999**
Mean	0.782	0.83	0.904	0.876	0.913	0.924	0.926	0.88	<u>0.931</u>	0.88	**0.963**

classification performance of the two variations of INSTINCT were compared, in terms of the accuracy and AUC scores. Then, we also analyzed the convergence of INSTINCT either when using FAP or GAP. Recall our hypothesis in Section 3 regarding the potentially slower, more spiky convergence expected when using GAP for STIC problems. That is due to the rough compression of the whole arbitrary-long time dimension into only a single, global value. To investigate this hypothesis, we first compared the convergence rates of the two variations of INSTINCT in terms of the number of epochs till convergence. In addition, in an attempt to estimate their convergence spikiness, the mean and standard deviation of the absolute value of the delta in test loss recorded in each epoch were computed. Assuming a total number of training epochs E, estimations are as follows $\mu_{spike} = \frac{\Sigma_{i=2}^{E}|loss_i - loss_{i-1}|}{E-1}$ and $\sigma_{spike} = \sqrt{\frac{\Sigma_{i=2}^{E}(|loss_i - loss_{i-1}| - \mu_{spike})^2}{E-1}}$, where larger values of μ_{spike} and σ_{spike} may indicate a potentially more spiky, inconsistent convergence.

Sensitivity analysis. Since the benchmark datasets are relatively small in terms of the number of data samples, there is a risk of overfitting them, especially when training a quite complex, optimized deep learning architecture. To mitigate the risk of overfitting in INSTINCT, several design decisions have been taken, including the use of an ensemble of classifiers as well as multiple steps of complexity reduction applied to the individual classifiers, as described in detail in Section 3. In addition, as previously noted, the hyperparameter values' optimization process as well as all the comparisons reported in the paper were conducted using 10-fold cross validation.

In this experiment, to yet increase the confidence that the proposed INSTINCT framework does not tend to overfit, we have also wanted to give a notion of how sensitive the model is to changes in its optimized hyperparameter values, inspired by [14]. For that purpose, we wanted to compare the classifications results of the k-best configurations of hyperparameter values of INSTINCT from the explored search space outlined in Table 2, for some $1 < k \in \mathbb{N}$. Specifically, $k = 5$ was selected, which means that the accuracy and AUC scores of INSTINCT when used with each of its 5-best performing configurations of hyperparameter values, were compared to each other as well as to state-of-the-art methods on the benchmark datasets.

5. Results

5.1. Experiment 1: Classification performance

Tables 3–4 summarize the accuracy and AUC scores of INSTINCT compared to state-of-the-art-methods for STIC [2,4,17,20,31] on the benchmark datasets, as well as on average. In addition, classification results of the three deep learning-based baselines employed on INSTINCT's ASTIMs representation are shown, where $base_{FF}$, $base_{LSTM}$, and $base_{CNN}$ stand for the feed-forward, LSTM, and CNN baselines described in the experimental setup respectively. Note that in Tables 3–4 best-performing methods in each dataset appear in bold, while second best scores are underlined.

Looking at Tables 3–4, it is clear that for the vast majority of datasets, and consequently on average, INSTINCT improves the classification performance over state-of-the-art, as well as over the employed baselines, in terms of both the accuracy and AUC scores. However, while INSTINCT indeed reached the most accurate classification results among the compared methods in all of

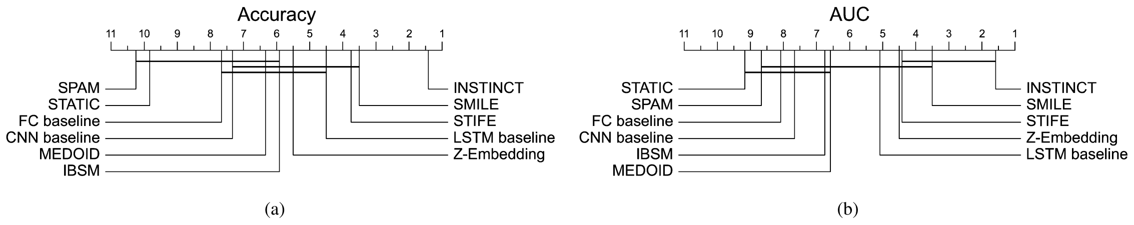


Fig. 6. Critical difference diagrams of the accuracy (a) and AUC (b) scores of INSTINCT compared to state-of-the-art methods, as well as to the three evaluated deep learning-based baselines, on the benchmark datasets.

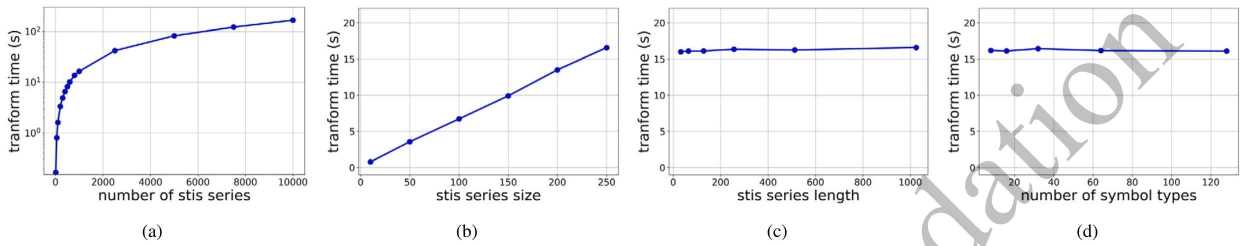


Fig. 7. Runtime of INSTINCT's Transform phase as a function of the number of STIs series (a), the series' size (b) and length (c), and the number of symbol types (d).

the individual datasets, it has not always been the only best-performing method. An example is the Blocks dataset, in which several existing methods [4,17,20,31] reached an equal accuracy score of 1.0 (which is, of course, the maximal possible accuracy). That is mainly due to the high predictive power of the pairwise temporal relations that appear in the Blocks dataset, which are explicitly encoded as features in the majority of previous methods and have usually been enough for perfect classification. That is, of course, not at all the case in the rest of the datasets, where more significant differences in accuracy have been reported.

As explained in detail in Section 4, to further test for statistical significance the Friedman test [8] was carried, followed by a pairwise post-hoc analysis using the Wilcoxon signed-rank test with $\alpha = 5\%$. The results are summarized in the critical difference diagrams shown in Fig. 6, in which (a) demonstrates the superior accuracy of INSTINCT compared to both state-of-the-art methods and the employed baselines on a highly significant level. In addition, note that the employed baselines reached reasonable accuracy scores, having the LSTM baseline almost as good as the best-performing state-of-the-art methods, and the CNN baseline just on-par as well in terms of statistical significance. These findings demonstrate the predictive power of INSTINCT's inception-based network beyond just the use of deep learning, as well as the potential in applying advanced deep-learning concepts from a wide range of fields (e.g., computer vision, natural language processing) to the proposed representation of STIs series as ASTIMs.

Finally, in terms of the AUC score, a statistically significant improvement was shown in INSTINCT over STATIC, SPAM, IBSM, MEDOID, Z-Embedding, and each of three deep learning-based baselines; while a non-significant improvement was reported compared to STIFE and SMILE, as shown in Fig. 6 (b). Nevertheless, it should be noted that the size of the sample which has been used for significance testing is quite small, since there are only few STIC benchmark datasets (Table 1). That is unlike related problems such as time series classification, where few tens of benchmark datasets are available online.

5.2. Experiment 2: Scalability analysis

After demonstrating that the proposed INSTINCT framework improves classification performance over state-of-the-art, in this experiment we present an empirical evaluation of INSTINCT's potential of scaling to large datasets. For that, we analyzed how the computation time of each of the two main phases of INSTINCT – i.e., Transform and Classify, is affected by the main four properties of STIC datasets, including 1) the number of STIs series (i.e., training samples), 2) the series' size and 3) length, and 4) the number of symbol types. That is based on the synthetically generated STIC datasets described in detail in Section 4.

Fig. 7 summarizes the results obtained for the Transform phase, through which the input STIs series are represented as ASTIMs in INSTINCT prior to being classified. In this figure (a–b), it is observed that runtime grows linearly with the number of STIs series (i.e., number of training samples) and the STIs series' size (i.e., number of STIs that they include). Meanwhile, the length of the STIs series and the number of symbol types seem to have no impact on this phase's runtime (c–d). That is since in the Transform phase, the raw series' STIs are embedded within their ASTIMs one STI at a time, regardless of their symbol types and time durations. This conforms with our hypothesis based on the analytical complexity analysis of the Transform phase provided in Subsection 3.1.

The results obtained for the Classify phase, through which the transformed ASTIMs are classified via INSTINCT's ensemble of deep CNN models, are presented in Fig. 8. In this figure a linear relationship is demonstrated between the network training time and the number of STIs series (a), their length (c), and the number of symbol types (d); which define the number of ASTIMs on which the network is trained as well as their dimensionality. Since the series' STIs had already been embedded within their ASTIMs during the preliminary Transform phase, varying the size of the STIs series, however, did not affect the network training time, as shown in Fig. 8 (b). In addition, recall our recommendation of using a stride of 10 time-units which equals to half the shortest filter of INSTINCT's

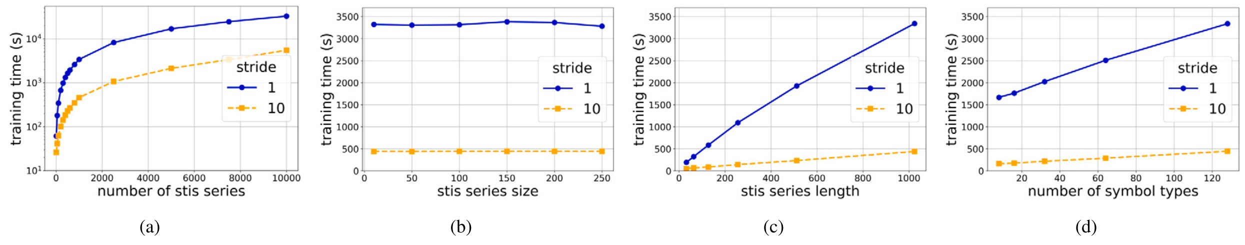


Fig. 8. Runtime of INSTINCT's Classify phase – i.e., the network training time, as a function of the number of STIs series (a), their size (b) and length (c), and the number of symbol types (d); for 200 epochs of training.

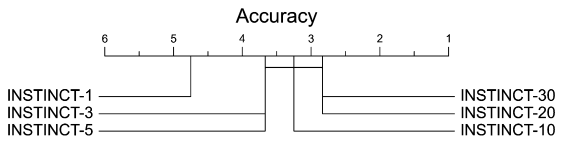


Fig. 9. Critical difference diagram of the accuracy score of INSTINCT for several ensemble sizes comprised of 1–30 classifiers.

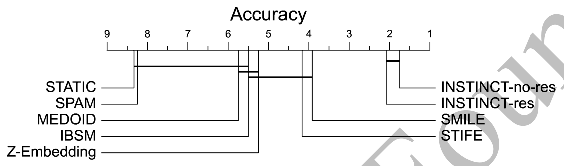


Fig. 10. Critical difference diagram of the accuracy score of INSTINCT with and without the use of residual connections, compared to state-of-the-art methods.

network, to reduce the model complexity and training time for datasets with thousands of time-units long STIs series (Subsection 3.2). In Fig. 8, we also show the network training time when using the proposed stride, reporting an expected, consistent reduction in runtime by a factor of 7–10 compared to the default stride of 1. This reduction is mainly due to the reduced number of performed convolutions in each residual block $i \in \{1, 2\}$ by a factor of 10^i, as described in Subsection 3.2.

Summing-up the results of the Transform and Classify phases, an overall empirical time complexity which is linear in each of the main properties of STIC datasets is observed for INSTINCT. These findings are encouraging as the current best-performing methods, i.e., STIFE and SMILE, have reported at least quadratic time complexities in both the number of STIs series, their size, and the number of symbol types (Subsection 2.2). In fact, INSTINCT's potential of scaling to large datasets is even more promising considering the trivial GPU parallelization of the network training, which has not been utilized on purpose in this experiment. In Appendix E, we also provide the empirical computational time of INSTINCT when being run on each of the real-world benchmark datasets (Table 1), beyond the extensive analysis conducted in this subsection on synthetic data.

5.3. *Experiment 3: Architecture study*

5.3.1. *Hyperparameters study*

The hyperparameter values of INSTINCT's ensemble of deep CNN models have been opted on the benchmark datasets, focusing on the trade-off between improved classification performance and reduced model complexity and training times. In Fig. 9–14 critical difference diagrams are presented, where each diagram compares multiple variations of INSTINCT with varied values of a specific hyperparameter. In these diagrams, running INSTINCT with the value of the hyperparameter tested set to v would be denoted by INSTINCT-v. If no such v is specified, then INSTINCT's final architecture, as introduced in Section 3, is referred.

Ensemble size. Random initialization of weight values of a neural network might result in a non-negligible variance in the classification results between successive runs. Hence, as described in Section 3, to increase the robustness of the proposed INSTINCT framework, an ensemble of multiple classifiers has been used, which are identical except for their randomly initialized weight values. Fig. 9 presents a critical difference diagram of the accuracy score of INSTINCT for several ensemble sizes, comprised of 1–30 classifiers. In this figure, only minor, non-statistically significant improvements are shown for ensembles of more than three classifiers, for which accuracy scores have been quite stable. Therefore, for the sake of minimizing training times, an ensemble size of three classifiers was used in INSTINCT, as illustrated in Fig. 3.

Residual connections. To evaluate the impact of the residual connections on the classification accuracy of INSTINCT, the framework was run once including the residual connections, and then without any such shortcut connection. The results are summarized in the critical difference diagram presented in Fig. 10, which compares the accuracy scores of these two variations of INSTINCT to state-of-the-art methods on the benchmark datasets.

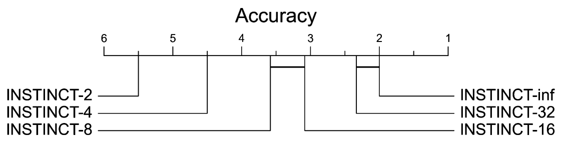


Fig. 11. Critical difference diagram comparing the accuracy score of INSTINCT for several bottleneck sizes $2 \leq b \leq 32$, as well as for $b = \infty$, which stands for the bottleneck layer being unused.

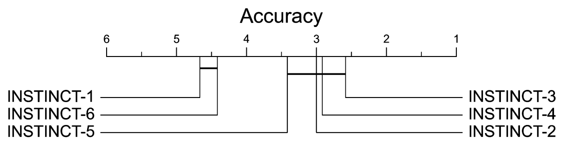


Fig. 12. Critical difference diagram of the accuracy score of INSTINCT depending on the network depth, i.e., the number of stacked inception modules.

While the two variations of INSTINCT outperformed state-of-the-art on a highly significant level, only a minimal difference in accuracy was reported between them. This outcome is, in fact, unsurprising since the proposed network architecture is quite shallow (i.e., comprised of only three stacked inception modules), and thus has lower likelihood to suffer from the problem of vanishing gradients, which skip connections aim to mitigate. Furthermore, in related problems in which the best-performing network architectures are typically much deeper, such as time series classification and computer vision, residual connections have mostly sped-up convergence while showing only minimal impacts on accuracy [14,39].

Bottleneck layer size. Another important feature of the proposed network, and in fact of the inception module, is the bottleneck layer. The bottleneck layer reduces the channels dimensionality of the input ASTIMs, which equals to the size of the symbols alphabet $|\Sigma|$. That is through applying $b << |\Sigma|$ one time-unit long convolutional filters with a stride of 1, for a bottleneck size b. Fig. 11 presents a critical difference diagram showing how the accuracy score of INSTINCT is affected by the bottleneck layer size b. That is, for $2 \leq b \leq 32$ as well as for $b = \infty$, which stands for the bottleneck layer being unused, in which case the original channels dimensionality of the ASTIMs is preserved.

In Fig. 11 a clear trend is observed, according to which higher accuracy rankings have been obtained for larger bottleneck sizes. That is mainly since the smaller the bottleneck b, the more significant the initial reduction in the dimensionality of the ASTIMs, and consequently in the model complexity, is. Thus, potentially forming high-biased models when using too small values of b. For larger values of b, on the other hand, accuracy scores have been consistently improving, which indicates that severe overfitting has not been reached even without the use of the bottleneck layer (i.e., when $b = \infty$). Yet, a non-significant difference in accuracy is observed already for $b = 32$.

Note that in practice, not using the bottleneck layer in this experiment stands for $b \approx 64$, since for all the benchmark datasets $|\Sigma| \leq 64$ (Table 1). That is except for the Pioneer dataset in which $|\Sigma| = 92$, however, accuracy scores have stabilized already for a very small bottleneck size of 8. Thus, using a bottleneck layer size of 32 typically reduces the channels dimensionality of the ASTIMs by roughly a factor of 2 while resulting in only a small reduction in accuracy. In that respect, it is also important to highlight that for datasets with much larger symbols alphabet sizes (e.g., few hundreds of symbol types), which do not present among the benchmark datasets, some of the explored bottleneck sizes might be too small and further exploration of larger bottleneck sizes may be of interest.

Network depth. One of the key hyperparameters of deep CNN architectures is the network depth, which enables the extraction of higher order, hierarchical features from spatial information, that are expected to further improve classification performance. Fig. 12 shows a critical difference diagram of INSTINCT's accuracy score depending on the network depth for 1–6 layers deep variations of INSTINCT, in terms of the number of stacked inception modules. Note that the networks explored in this experiment are quite shallow compared to state-of-the-art deep CNN architectures for computer vision, for example. That is mainly due to the relatively small size of STIC benchmark datasets, which include only several hundreds of data samples. Thus, strictly limiting the explorable complexity of the models, aiming to avoid the problem of overfitting.

In Fig. 12, an unsurprising yet significant improvement is observed in accuracy when increasing the network depth by stacking two or three inception modules, through which longer patterns, possibly at higher abstraction levels, are expected to be detected within the input ASTIMs. Further increasing the network depth, however, resulted in reduced accuracy scores. While stacking 4–5 modules showed only a slight decrease in accuracy, for a deeper architecture comprised of six stacked inception modules, a significant reduction was reported. That is potentially due to overfitting few of the benchmark datasets, a trend which is expected to intensify for even deeper networks. Therefore, as presented in Section 3, the final architecture of INSTINCT is comprised of three stacked inception modules, that has reached the best classification performance at a reasonable model complexity.

Filter length. Another important hyperparameter of deep CNN architectures in general, and inception-based networks in particular, is the length of convolutional filters employed in each of the network's convolutional layers. To analyze how the length of the applied filters affects the accuracy of INSTINCT, the framework was run multiple times using only a single, uniform length of filters throughout the network. That is, for filter lengths corresponding to the powers of two from 8 to 64. We highlight that these filters

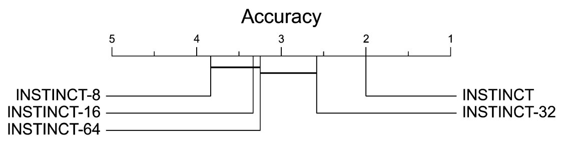


Fig. 13. Critical difference diagram of the accuracy score of INSTINCT for several individual filter lengths, as well as of the final architecture of INSTINCT which combines 20, 30, and 40 time-units long filters via the inception module.

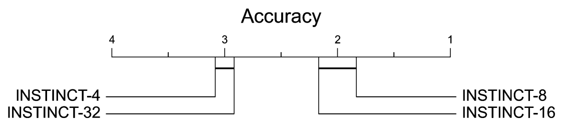


Fig. 14. Critical difference diagram of the accuracy score of INSTINCT depending on the number of convolutional filters applied in each of the inception module's convolutions.

are quite long compared to convolutional filters that are typically employed in the field of computer vision. That is mainly due to the one less spatial dimension of ASTIMs compared to images, which allows the exploration of much longer filters that may indeed be beneficial for the detection of prolonged patterns within arbitrary long STIs series data.

The results of running INSTINCT with each of the specified uniform filter lengths (i.e., 8, 16, 32, and 64), as well as of the final architecture of INSTINCT which combines multiple filter lengths via the inception module, are summarized in the critical difference diagram presented in Fig. 13. First, looking at the individual filter lengths only, a significant improvement in accuracy was observed when increasing the filter length of INSTINCT from 8 to 16, or even to 32 time-units. This significant improvement has been predictable, since longer filters are known to better detect longer patterns which potentially reside within the input ASTIMs. Further doubling the filter length, however, showed a reduction in accuracy, yet not significant, for 64 time-units long filters. That is mainly due to the much increased model complexity, which resulted in a somewhat higher tendency of the model to overfit, and as a consequence not generalize well enough to unseen data.

These findings have led us to the final design of INSTINCT's network presented in Section 3, which combines three filter lengths of 20, 30, and 40 time-units, surrounding ($\sim \pm 10$ time-units) the best performing individual filter length of 32. That is via the inception module, which enables to simultaneously apply filters of varied lengths to the input ASTIMs. As shown in Fig. 13, this architecture has further improved accuracy over the best performing individual filter length, reporting a statistically significant difference. This demonstrates the power of applying inception-based convolutional neural networks to STIs series data, where the high flexibility of the inception module is beneficial for well-classifying data samples of various lengths and densities.

Number of filters. At last, after determining the filter lengths to be used in the inception module, we also wanted to investigate the effect of the number of filters applied in each convolution operation in INSTINCT on the framework's accuracy. Note that the number of filters must be set with great caution to the risk of overfitting. That is since increasing the number of filters by a factor f, results in an increased number of trainable parameters in each convolutional layer, as well as in the final softmax layer, by roughly the same factor. Fig. 14 shows a critical difference diagram of the accuracy score of INSTINCT with a varied number of filters corresponding to the powers of two from 4 to 32. While increasing the number of filters from 4 to either 8 or 16 filters resulted in a statistically significant improvement in accuracy, only a minimal difference was reported between them. Running INSTINCT with a further doubled number of convolutional filters (i.e., 32), however, resulted in less accurate classification results, potentially due to the higher tendency of the model to overfit. Therefore, 8 filters were selected to be applied in each convolution operation in INSTINCT. That is mainly due to the significantly reduced complexity of the model, compared to the use of 16 filters. Note that this stands for an overall number of $8 \cdot 4 = 32$ filters per inception module, due to the four convolutions applied in parallel, as explained in detail in Subsection 3.2.

5.3.2. *FAP vs GAP*

In this experiment, we further investigated the proposed usage of flattening and average pooling (FAP) over global average pooling (GAP) in INSTINCT, for the conversion of the inception module's output into a feature vector. That is, in terms of both the classification performance and convergence aspects. First, Fig. 15 presents critical difference diagrams of the accuracy and AUC scores of INSTINCT either when using FAP or GAP, compared to state-of-the-art methods on the benchmark datasets. While the two variations of INSTINCT showed improved classification performance over state-of-the-art in terms of either evaluation metric, a high significance level of improvement in accuracy was reported only when using FAP. Yet, despite the indeed improved classification performance obtained for FAP compared to the use of GAP, as hypothesized, the differences between the two variations of INSTINCT were not statistically significant on the given benchmark.

To further analyze the convergence of INSTINCT either when using FAP or GAP, we compared the convergence rates of the two variations of INSTINCT in terms of the number of epochs till convergence, for at most 200 epochs of training. The results are summarized in Fig. 16, which compares the distribution of convergence rates of the evaluated variations of INSTINCT for each dataset, over 10-fold cross-validation. While early convergence has not been reached in either variation along the first 200 epochs of training in the Auslan2 dataset, looking at Fig. 16 it is quite clear to see that in the rest of datasets convergence has been much faster when using FAP, by an average factor of 54. That is, while also reporting typically lower variance among the different folds of each

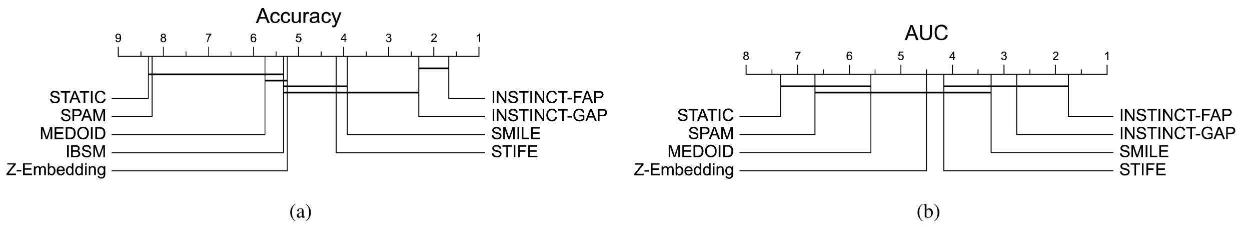


Fig. 15. Critical difference diagrams of the accuracy and AUC scores of INSTINCT either when using flattening and average pooling (FAP) as proposed in the paper, or a global average pooling layer (GAP); compared to state-of-the-art methods.

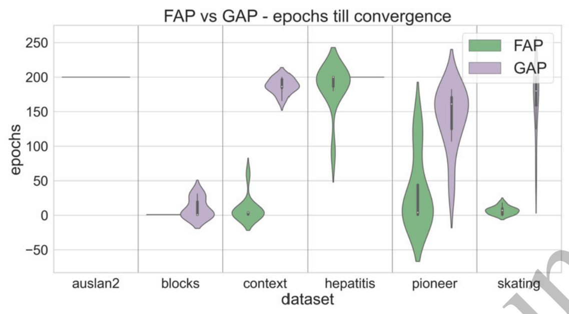


Fig. 16. Violin plot comparing the distribution of convergence rates of INSTINCT – either when using FAP or GAP, on the benchmark datasets over 10-fold cross-validation.

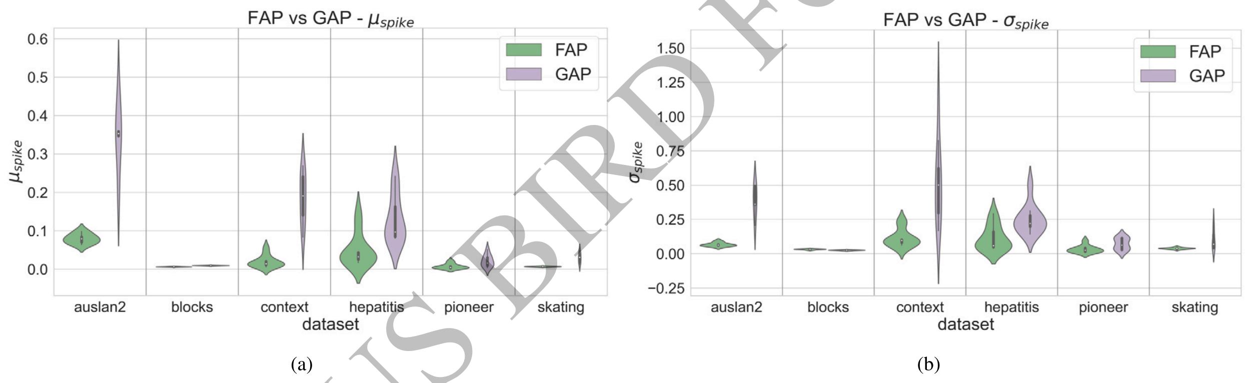
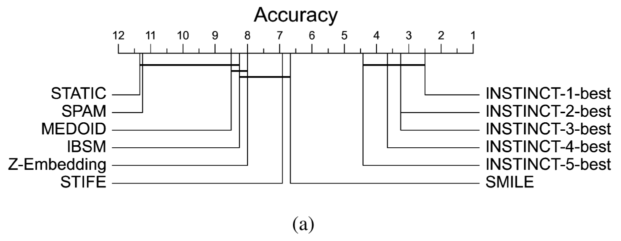


Fig. 17. Violin plots comparing the distribution of convergence spikiness estimation metrics μ_{spike} (a) and σ_{spike} (b) of INSTINCT – either when using FAP or GAP, on the benchmark datasets over 10-fold cross-validation.

dataset. To test for statistical significance, we carried out a t-test as well as a Wilcoxon signed-rank test, resulting in p-value $\approx 1.2e-7$ in the former and p-value $\approx 5.4e-7$ in the latter.

Finally, we also attempted to estimate the degree of convergence spikiness of each of the two variations of INSTINCT. That is, in terms of the convergence spikiness estimation metrics μ_{spike} and σ_{spike} introduced in Subsection 4.2.3, which stand for the mean and standard deviation of the absolute value of the delta in test loss recorded along the network training epochs respectively. The results are summarized in Fig. 17, which compares the distribution of μ_{spike} (a) and σ_{spike} (b) values obtained for INSTINCT either when using FAP or GAP, for each dataset over 10-fold cross-validation.

Fig. 17 (a) demonstrates that lower values of μ_{spike} have been typically reported when using FAP in all of the benchmark datasets, by an average factor of 5.42, while also showing somewhat reduced variance among folds. In Fig. 17 (b) a similar trend is observed in the distribution of σ_{spike} values. That is except for the Blocks datasets, in which the use of GAP resulted in slightly lower values of σ_{spike}. Yet, overall, a reduction factor of 3.53 has been reported on average in σ_{spike} when using FAP compared to the use of GAP. Moreover, in terms of both μ_{spike} and σ_{spike}, the use of FAP resulted in a statistically significant improvement over GAP, reporting p-value $< 1.1e-5$ and p-value $< 1e-9$ in a t-test and a Wilcoxon signed-rank test respectively, which compared the two populations. Thus, strengthening the assumptions made in Subsection 4.2.3 regarding the potentially slower, more spiky convergence of INSTINCT when using GAP for STIC compared to the use of FAP. In Appendix D, we exemplify how the behavior of INSTINCT's loss curve along the training process – either when using FAP or GAP, is reflected in μ_{spike} and σ_{spike}. That is, when processing the Context and Pioneer datasets, on which the largest and smallest differences in μ_{spike} and σ_{spike} have been reported between the two variations of INSTINCT respectively, as shown in Fig. 17.



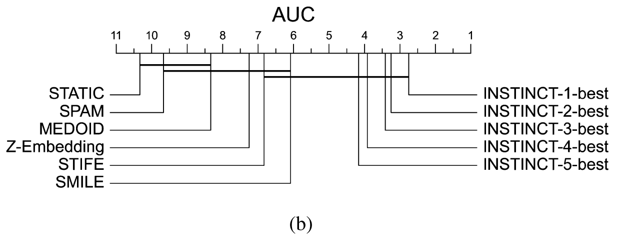


Fig. 18. Critical difference diagrams of the accuracy and AUC scores of INSTINCT when used with each of its 5-best configurations of hyperparameter values, compared to state-of-the-art methods.

5.3.3. Sensitivity analysis

At last, we wanted to investigate how sensitive INSTINCT is to small changes in its optimized hyperparameter values. For that, the accuracy and AUC scores of INSTINCT when used with each of its 5-best configurations of hyperparameter values, were compared to each other as well as to state-of-the-art methods. The results are summarized in the critical difference diagrams presented in Fig. 18, in which only minor, non-significant differences are observed among all the evaluated configurations of INSTINCT. In addition, in terms of their statistical significance compared to state-of-the-art, it is quite clear to see that they are all virtually the same. These findings demonstrate a low sensitivity of INSTINCT to small changes in its optimized hyperparameter values, which, along with the use of 10-fold cross-validation, increase the confidence that the proposed framework does not tend to overfit.

6. Discussion and conclusions

In this paper we introduced INSTINCT – a novel deep learning-based framework for the classification of series of symbolic time intervals (STIC). INSTINCT exhibits an almost fully information-preserving transformation of raw STIs series into real matrices through the concept of ASTIMs. The proposed transformation enables to bridge between the field of STIs data processing and several other, more intensely researched fields such as computer vision, natural language processing, and time series analysis. Thus, opening the door for significant future work of bringing the forefront of research within these fields into a large variety of valuable, yet immaturely researched problems involving STIs data, e.g., STIs series clustering, classification, embedding etc.

In INSTINCT, the transformed ASTIMs are classified via a novel ensemble of three deep inception-based classifiers for STIC. The evaluation has demonstrated that INSTINCT significantly improves state-of-the-art classification accuracy on the real-world STIC benchmark datasets. That is, while also reporting high scalability – showing a linear dependency between its overall runtime and each of the main properties of STIs series datasets. Finally, an extensive architecture study of INSTINCT has been conducted, focusing on the trade-off between improved classification performance and reduced model complexity and training times, as well as on the analysis of INSTINCT's sensitivity and convergence characteristics. For future work, we would like to build upon the proposed representation of STIs series as ASTIMs to investigate the integration of recent advancements made in the related research fields of time series analysis (e.g., ROCKET and its variants) and natural language processing (e.g., self-attention), into STIs series data processing. In addition, we would also like to develop more flexible and efficient representations for very long STIs series that are usually rather sparse.

CRediT authorship contribution statement

Omer David Harel: Conceptualization, Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft. **Robert Moskovitch:** Conceptualization, Funding acquisition, Methodology, Resources, Supervision, Visualization, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Code and data are available online on GitHub (link is provided in the paper).

Acknowledgements

This research was partially funded by a grant of the Israeli Ministry of Science and Technology (grant 8760441). Omer David Harel was also funded by the Darom-Lachish scholarship of Kreitman School of Advanced Graduate Studies at Ben Gurion University (no. 1955129).

Appendix A. Real-world data

In this appendix, detailed information is provided on the real-world STIC benchmark datasets [25,30] which have been used for the evaluation of INSTINCT.

- **Auslan2.** STIs have been derived from the publicly available Australian Sign Language dataset in the UCI repository, in which each sample, i.e., STIs series, represents a single word.
- **Blocks.** STIs represent visual primitives that have been drawn from videos of a human hand stacking colored blocks. Each STIs series stands for one of eight scenarios, including either atomic actions (e.g., *move-right*) or complete scenarios (e.g., *assemble*).
- **Context.** STIs have been extracted from categorical and numerical data which describe the context of a mobile device carried by humans in different scenarios. Each STIs series represents one of five such scenarios (e.g., *meeting* or *street*).
- **Hepatitis.** STIs series describe series of tests conducted to patients suffering from either Hepatitis *B* or *C* over a time period of 10 years.
- **Pioneer.** STIs have been derived from the Pioneer-1 dataset in the UCI repository, which includes data collected from the Pioneer-1 mobile robot's sensor readings. Each STIs series describes one of the robot's three moving scenarios, i.e., either *gripper*, *move*, or *turn*.
- **Skating.** STIs have been derived from fourteen-dimensional numerical time series, which describe both the muscle activity and leg position of six professional In-Line Speed Skaters during controlled tests. Each STIs series represents a complete movement cycle.

Appendix B. Raw STIs series reconstruction from ASTIMs

Under the assumption that multiple STIs which have the same symbol do not overlap, raw STIs series can be fully reconstructed from their transformed ASTIMs as follows.

Algorithm 1 Raw STIs Series reconstruction from an ASTIM.

```
Input: Let $\Sigma$ be an alphabet of symbols, and an ASTIM $A^S \in \mathbb{R}^{|\Sigma| \times |S|}$.
Output: $S$ – reconstructed raw STIs series
1: $S \leftarrow \emptyset$
2: $start \leftarrow -1,\ finish \leftarrow -1$
3: for $\sigma \in \Sigma$ do
4:     for $t \leftarrow 0$ to $|S|$ do
5:         if $A^S_{\sigma,t} > 0 \wedge (t = 0 \vee A^S_{\sigma,t-1} = 0)$ then
6:             $start \leftarrow t$
7:         else if $A^S_{\sigma,t} = 0 \wedge t > 0 \wedge A^S_{\sigma,t-1} > 0$ then
8:             $finish \leftarrow t$
9:             $I \leftarrow (\sigma, start, finish)$
10:            $S \leftarrow S \cup \{I\}$
11:            $start \leftarrow -1,\ finish \leftarrow -1$
12:        end if
13:    end for
14: end for
15: return $S$
```

Appendix C. Configurations of synthetic datasets for scalability analysis

See Table C.5 below.

Appendix D. FAP vs GAP – Illustration of convergence spikiness estimation metrics

In this appendix, we aim to give a notion of how the convergence spikiness estimation metrics μ_{spike} and σ_{spike} introduced in Subsection 4.2.3, are reflected in INSTINCT's loss curve either when using FAP or GAP. For that, we use the Context and Pioneer datasets, on which the largest and smallest differences in μ_{spike} and σ_{spike} have been recorded between the two evaluated variations of INSTINCT respectively (Fig. 17).

Fig. D.19 depicts the loss curve of INSTINCT along 200 epochs of training, while Table D.6 lists the respective values of μ_{spike} and σ_{spike} obtained in the Context and Pioneer datasets, either when using FAP or GAP. In the Context dataset, using FAP in INSTINCT resulted in an eight times smaller value of μ_{spike}, and about four times smaller value of σ_{spike} compared to the use of GAP. This is indeed reflected in Fig. D.19 (a), which shows the much smoother loss curve of INSTINCT when using FAP compared to the rather spiky curve obtained when using GAP. In the Pioneer dataset, on the other hand, both variations of INSTINCT resulted in much smaller values of μ_{spike} and σ_{spike}, having FAP reporting only 1.5–2 times lower measurements compared to the use of GAP. That is due to the more consistent convergence of both variations of INSTINCT observed in Fig. D.19 (b), in which only several moderate spikes in loss have occurred when using GAP.

Table C.5
Input configurations for the generation of synthetic data used for INSTINCT's scalability analysis, where n stands for the number of STIs series, m and l represent the series size and length respectively, and $|\Sigma|$ stands for the symbols alphabet size.

examined property	n	m	l	$\|\Sigma\|$
n	10	256	1024	128
	50			
	100			
	200			
	300			
	400			
	500			
	600			
	800			
	1000			
	2500			
	5000			
	7500			
	10000			
m	1024	10	1024	128
		50		
		100		
		150		
		200		
		250		
l	1024	256	32	128
			64	
			128	
			256	
			512	
			1024	
$\|\Sigma\|$	1024	256	1024	8
				16
				32
				64
				128

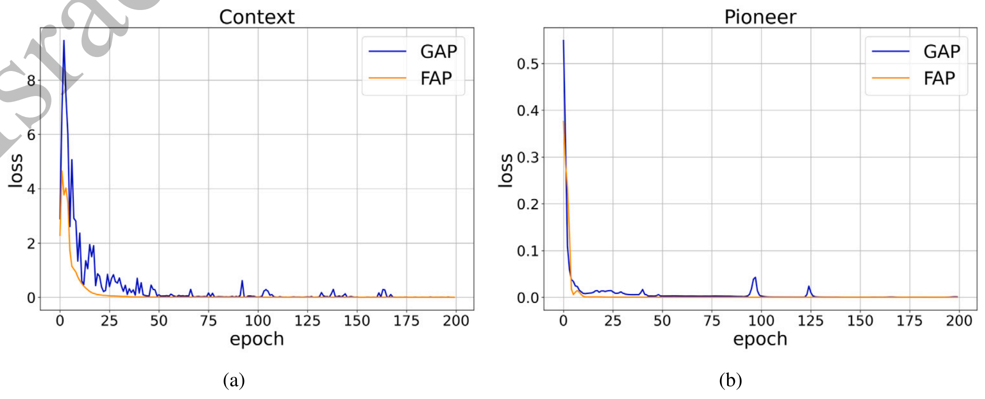


Fig. D.19. Illustrations of INSTINCT's loss curve either when using FAP or GAP on the Context and Pioneer datasets.

Table D.6
μ_{spike} and σ_{spike} values obtained for INSTINCT in the Context and Pioneer datasets, either when using FAP or GAP.

Dataset	μ_{spike}		σ_{spike}	
	FAP	GAP	FAP	GAP
Context	0.022	0.177	0.100	0.469
Pioneer	0.002	0.004	0.013	0.022

Table E.7
INSTINCT's Transform time, network epoch time – either when running on CPU or GPU, and the mean number of epochs till convergence, in each of the real-world STIC datasets.

Dataset	Transform time (s)	Network epoch time (s)		Mean #epochs till convergence
		CPU	GPU	
Auslan2	0.156	0.025	0.018	200
Blocks	0.087	0.067	0.020	1
Context	1.117	1.193	0.022	9
Hepatitis	3.561	5.379	1.063	185.3
Pioneer	0.547	0.061	0.023	34
Skating	1.495	5.328	1.046	7

Appendix E. INSTINCT's runtime on real-world datasets

Table E.7 summarizes INSTINCT's Transform time and network epoch time, either when running on CPU or a GTX 1660 Ti GPU, as well as the mean number of epochs till convergence, in each of the real-world STIC datasets (Table 1). As described in Subsection 4.2, INSTINCT's network has been trained for at most 200 epochs. Thus, the total time of the Classify phase, i.e., the network training time, is bounded by two hundred times the epoch time, although in the majority of datasets convergence has been typically reached much faster, as shown in Table E.7 as well as in Fig. 16.

References

[1] J.F. Allen, Maintaining knowledge about temporal intervals, Commun. ACM 26 (1983) 832–843, https://doi.org/10.1145/182.358434.
[2] J. Ayres, J. Flannick, J. Gehrke, T. Yiu, Sequential pattern mining using a bitmap representation, in: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2002, pp. 429–435.
[3] I. Batal, H. Valizadegan, G.F. Cooper, M. Hauskrecht, A temporal pattern mining approach for classifying electronic health record data, ACM Trans. Intell. Syst. Technol. 4 (2013) 1–22.
[4] L. Bornemann, J. Lecerf, P. Papapetrou, Stife: a framework for feature-based classification of sequences of temporal intervals, in: International Conference on Discovery Science, Springer, 2016, pp. 85–100.
[5] A. Dempster, F. Petitjean, G.I. Webb, Rocket: exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (2020) 1454–1495, https://doi.org/10.1007/s10618-020-00701-z.
[6] A. Dempster, D.F. Schmidt, G.I. Webb, Minirocket: a very fast (almost) deterministic transform for time series classification, in: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, 2021, pp. 248–257.
[7] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[8] M. Friedman, A comparison of alternative tests of significance for the problem of m rankings, Ann. Math. Stat. 11 (1940) 86–92, https://doi.org/10.1214/aoms/1177731944.
[9] O. Harel, R. Moskovitch, Complete closed time intervals-related patterns mining, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2021, pp. 4098–4105.
[10] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[11] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Min. Knowl. Discov. 28 (2014) 851–881, https://doi.org/10.1007/s10618-013-0322-1.
[12] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: International Conference on Machine Learning, PMLR, 2015, pp. 448–456.
[13] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (2019) 917–963, https://doi.org/10.1007/s10618-019-00619-1.
[14] H. Ismail Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.A. Muller, F. Petitjean, Inceptiontime: finding alexnet for time series classification, Data Min. Knowl. Discov. 34 (2020) 1936–1962, https://doi.org/10.1007/s10618-020-00710-y.
[15] D.P. Kingma, J. Ba, Adam: a method for stochastic optimization, arXiv preprint, arXiv:1412.6980, 2014.
[16] O. Kostakis, P. Papapetrou, J. Hollmén, Artemis: assessing the similarity of event-interval sequences, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2011, pp. 229–244.
[17] A. Kotsifakos, P. Papapetrou, V. Athitsos, Ibsm: interval-based sequence matching, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 596–604.
[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Commun. ACM 60 (2017) 84–90.
[19] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (1998) 2278–2324.
[20] Z. Lee, Š. Girdzijauskas, P. Papapetrou, Z-embedding: a spectral representation of event intervals for efficient clustering and classification, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2020, pp. 710–726.

[21] J. Lines, S. Taylor, A. Bagnall, Hive-cote: the hierarchical vote collective of transformation-based ensembles for time series classification, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 1041–1046.
[22] L. Liu, S. Wang, B. Hu, Q. Qiong, J. Wen, D.S. Rosenblum, Learning structures of interval-based bayesian networks in probabilistic generative model for human complex activity recognition, Pattern Recognit. 81 (2018) 545–561, https://doi.org/10.1016/j.patcog.2018.04.022.
[23] R. Liu, Y. Li, L. Tao, D. Liang, H.T. Zheng, Are we ready for a new paradigm shift? A survey on visual deep mlp, Patterns 3 (2022) 100520.
[24] B. Lucas, A. Shifaz, C. Pelletier, L. O'Neill, N. Zaidi, B. Goethals, F. Petitjean, G.I. Webb, Proximity forest: an effective and scalable distance-based classifier for time series, Data Min. Knowl. Discov. 33 (2019) 607–635, https://doi.org/10.1007/s10618-019-00617-3.
[25] F. Mörchen, D. Fradkin, Robust mining of time intervals with semi-interval partial order patterns, in: Proceedings of the 2010 SIAM International Conference on Data Mining, SIAM, 2010, pp. 315–326.
[26] R. Moskovitch, Multivariate temporal data analysis - a review, Wiley Interdiscip. Rev. Data Min. Knowl. Discov. 12 (2022) e1430.
[27] R. Moskovitch, Y. Shahar, Classification of multivariate time series via temporal abstraction and time intervals mining, Knowl. Inf. Syst. 45 (2015) 35–74, https://doi.org/10.1007/s10115-014-0784-5.
[28] R. Moskovitch, C. Walsh, F. Wang, G. Hripcsak, N. Tatonetti, Outcomes prediction via time intervals related patterns, in: 2015 IEEE International Conference on Data Mining, IEEE, 2015, pp. 919–924.
[29] P. Novitski, C.M. Cohen, A. Karasik, G. Hodik, R. Moskovitch, Temporal patterns selection for all-cause mortality prediction in t2d with anns, J. Biomed. Inform. 134 (2022) 104198, https://doi.org/10.1016/j.jbi.2022.104198.
[30] D. Patel, W. Hsu, M.L. Lee, Mining relationships among interval-based events for classification, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, 2008, pp. 393–404.
[31] J. Rebane, I. Karlsson, L. Bornemann, P. Papapetrou, Smile: a feature-based temporal abstraction framework for event-interval sequence classification, Data Min. Knowl. Discov. 35 (2021) 372–399, https://doi.org/10.1007/s10618-020-00719-3.
[32] L. Sacchi, C. Larizza, C. Combi, R. Bellazzi, Data mining with temporal abstractions: learning rules from time series, Data Min. Knowl. Discov. 15 (2007) 217–247, https://doi.org/10.1007/s10618-007-0077-7.
[33] P. Schäfer, The boss is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (2015) 1505–1530, https://doi.org/10.1007/s10618-014-0377-7.
[34] E. Sheetrit, N. Nissim, D. Klimov, Y. Shahar, Temporal probabilistic profiles for sepsis prediction in the icu, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 2961–2969.
[35] A. Shifaz, C. Pelletier, F. Petitjean, G.I. Webb, Ts-chief: a scalable and accurate forest algorithm for time series classification, Data Min. Knowl. Discov. 34 (2020) 742–775, https://doi.org/10.1007/s10618-020-00679-8.
[36] G. Shitrit, N. Tractinsky, R. Moskovitch, Visualization of frequent temporal patterns in single or two populations, J. Biomed. Inform. 134 (2022) 104169, https://doi.org/10.1016/j.jbi.2022.104169.
[37] M. Shokoohi-Yekta, B. Hu, H. Jin, J. Wang, E. Keogh, Generalizing dtw to the multi-dimensional case requires an adaptive approach, Data Min. Knowl. Discov. 31 (2017) 1–31, https://doi.org/10.1007/s10618-016-0455-0.
[38] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, arXiv preprint, arXiv:1409.1556, 2014.
[39] C. Szegedy, S. Ioffe, V. Vanhoucke, A.A. Alemi, Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-First AAAI Conference on Artificial Intelligence, 2017.
[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
[41] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.
[42] C.W. Tan, A. Dempster, C. Bergmeir, G.I. Webb, Multirocket: multiple pooling operators and transformations for fast and effective time series classification, Data Min. Knowl. Discov. 36 (2022) 1623–1646, https://doi.org/10.1007/s10618-022-00844-1.
[43] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: a strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1578–1585.
[44] H. Xing, Z. Xiao, D. Zhan, S. Luo, P. Dai, K. Li, Selfmatch: robust semisupervised time-series classification with self-distillation, Int. J. Intell. Syst. 37 (2022) 8583–8610.
[45] J. Yu, Y. Rui, D. Tao, Click prediction for web image reranking using multimodal sparse coding, IEEE Trans. Image Process. 23 (2014) 2019–2032.
[46] J. Yu, M. Tan, H. Zhang, Y. Rui, D. Tao, Hierarchical deep click feature prediction for fine-grained image recognition, IEEE Trans. Pattern Anal. Mach. Intell. 44 (2019) 563–578.
[47] J. Zhang, Y. Cao, Q. Wu, Vector of locally and adaptively aggregated descriptors for image feature representation, Pattern Recognit. 116 (2021) 107952.
[48] L. Zhang, S. Huang, W. Liu, D. Tao, Learning a mixture of granularity-specific experts for fine-grained categorization, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 8331–8340.
[49] X. Zhang, Y. Gao, J. Lin, C.T. Lu, Tapnet: multivariate time series classification with attentional prototypical network, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2020, pp. 6845–6852.