Deep semi-supervised clustering for multi-variate time-series

Dino Ienco, Roberto Interdonato

Neurocomputing 516 (2023) 36–47

Contents lists available at ScienceDirect

Neurocomputing

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

Deep semi-supervised clustering for multi-variate time-series

Dino Ienco [a,*], Roberto Interdonato [b]

[a] INRAE, UMR TETIS, Montpellier, France
[b] CIRAD, UMR TETIS, Montpellier, France

ARTICLE INFO

Article history:
Received 24 August 2021
Revised 11 August 2022
Accepted 15 October 2022
Available online 20 October 2022
Communicated by Zidong Wang

Keywords:
Multi-variate time series
Clustering
Semi-Supervised
Constrained clustering

ABSTRACT

Huge amount of data are nowadays produced by a large and disparate family of sensors, which typically measure multiple variables over time. Such rich information can be profitably organized as multivariate time-series. Collect enough labelled samples to set up supervised analysis for such kind of data is challenging while a reasonable assumption is to dispose of a limited background knowledge that can be injected in the analysis process. In this context, semi-supervised clustering methods represent a well suited tool to get the most out of such reduced amount of knowledge. With the aim to deal with multivariate time-series analysis under a limited background knowledge setting, we propose a semi-supervised (constrained) deep embedding time-series clustering framework that exploits knowledge supervision modeled as Must- and Cannot-link constraints. More in detail, our proposal, named conDetSEC (constrained Deep embedding time SEries Clustering), is based on Gated Recurrent Units (GRUs) with the aim to explicitly manage the temporal dimension associated to multi-variate time series data. conDetSEC implements a procedure in which an embedding generation step is combined with a clustering refinement step. Both steps exploit the small amount of available knowledge provided by Must- and Cannot-link constraints. More specifically, during the data embedding generation the constraints are used by jointly optimizing the network parameters via both unsupervised and semi-supervised tasks, while at the refinement step they are used in conjunction with the goal to stretch the embedding manifold towards the clustering centroids to recover a more clear cluster structure. Experimental evaluation on real-world benchmarks coming from diverse domains has highlighted the effectiveness of our proposal in comparison with state-of-the-art unsupervised and semi-supervised time-series clustering methods.

1. Introduction

Nowadays, the continuous acquisition of massive data has become an essential part in processes at the basis of several application domains, such as agriculture, biochemistry and human health. Such data is acquired through the use of domain-specific sensors (e.g., remote sensors, biochemical sensors, wearable devices) that are able to produce data streams including the tracking of multiple features over time. When there's the need to computationally analyze such data, the natural way to model these streams is into multivariate time-series. These data structures are gaining increasing interest, and several methods have been proposed in recent years that address tasks such as classification [36,17,39] and forecasting [24,26,22,37] of multivariate time-series.

However, the effective clustering of multivariate time-series, though being a central problem in a plethora of practical tasks, remains an open problem [19]. In real-world scenarios, labeling data is an expensive task in terms of both time and resources, regardless of the application domain. That's why developing advanced unsupervised and semi-supervised clustering approaches is a priority in this context [19], also considering the fact that these methods allow to characterize multivariate time-series data without taking into account any apriori knowledge.

In this domain, like in many others, deep learning approaches have recently proven to be generally more effective than classic data science approaches. That is, the ability of such neural architectures to learning suitable data representation have a major impact in optimizing the task at hand, and this is even more evident in the multi-variate time series context, where the need to handle the time dimension adds complexity to the already challenging subject of multivariate data [29,11].

A deep learning based unsupervised clustering method for multivariate time series has been recently proposed in [16], which

* Corresponding author.
E-mail addresses: dino.ienco@inrae.fr (D. Ienco), roberto.interdonato@cirad.fr (R. Interdonato).

https://doi.org/10.1016/j.neucom.2022.10.033
0925-2312/

exploits a recurrent autoencoder integrating attention and gating mechanisms in order to produce effective embeddings of the input data. However, in practical contexts, it would be beneficial to exploit some amount of available knowledge: even when this quantity is too small to be used for the training of a proper classification model, it would still be valuable to use it to steer the clustering process, which is clearly not possible where going for a completely unsupervised method. For this reason, in this work we propose a new method especially tailored for constrained clustering of multi-variate time series data, namely conDetSEC, that extends the approach in [16] to the semi-supervised setting, by taking into account available (small amount of) background knowledge under the shape of *Must-link* (ML) and *Cannot-link* (CL) constraints.

To this end, we propose a neural network architecture based on Gated Recurrent Units (GRUs), in which the clustering process is based on a two steps procedure involving an embedding generation and a clustering refinement step, both exploiting the small amount of available knowledge provided by ML and CL constraints. More specifically, during the data embedding generation the constraints are used by jointly optimizing the network parameters via both unsupervised and semi-supervised tasks, while at the refinement step they are used in conjunction with the objective to stretch the embedding manifold towards the clustering centroids to recover a more clear cluster structure.

With the aim to assess the behavior of our framework, we provide an experimental analysis on six real-world time-series benchmarks coming from different domains, that shows the effectiveness and flexibility of our approach. The results of this analysis clearly show how conDetSEC is able to exploit increasing quantities of labeled data, with clustering results that always improve with percentage of available supervision.

The rest of the paper is structured as follows: Section 2 discusses related work, the conDetSEC framework is introduced in Section 3, Section 4 presents experimental evaluation, while Section 5 concludes.

2. Related work

Time series clustering is a problem that has been largely addressed in literature in its univariate form [21], but that remains challenging while coming to the multi-variate case. Focusing on completely unsupervised approaches, several extensions of well known techniques have been proposed [3,35,6,10]. More recent approaches include Markov Random fields based techniques for subsequence clustering [14], deep learning based approaches for agglomerative clustering of video data [30] and for the clustering of variable-length time series [29]. As regards constrained clustering, an insight into how different single-algorithm and ensemble formulations can be adapted to the time-series case (i.e., by introducing Dynamic Time Warping as a more suitable distance measure and DBA (DTW Barycenter Averaging) method to compute cluster centroids) is presented in [20]. The analysis highlights how k-means based approaches are less suitable for the task, with respect to spectral based and declarative ones. In [32] a semi-supervised clustering methods for time series data is proposed, namely COBRAS^{TS}, which is an extension of the COBRAS method previously proposed by the same authors [31]. The main idea behind this method is that of introducing *super-instances*, i.e., sets of items that represent an intermediate step between the original itemset and the final clustering. The process begins with all the items belonging to the same super-instance, which is then decomposed in smaller ones through an iterative refinement process, which takes into account must-link and cannot-link constraints. Then the new super-instances are reassigned to new clusters. COBRAS^{TS} extends the original algorithm by introducing distance measures (DTW and shape-based) and refinement algorithms (spectral k-Shape clustering) suitable for time series. Recently, Fontes et al. [12] proposed a framework involving fuzzy c-means based constrained clustering of time series in a pattern reconciliation context. The technique of data reconciliation is oriented to the minimization of measurement errors in the data by imposing physical constraints associated with the production system (e.g., mass and energy balances). That is, in this case the method deals with soft constrains related to the physical properties of the cluster members in a specific domain, and not with the classic Must-link and Cannot-link constraints generally adopted in data science methods. Summarizing, constrained clustering of multi-variate time series is still an open problem, as also recently observed by the authors in [12]. This represents a major motivation behind the proposal of the conDetSEC framework, which could fill this gap in literature, thus contributing in addressing this task in a plethora of practical problems in different domains.

3. Methodology

In this section we introduce a new constrained clustering framework especially tailored to manage multi-variate time series data referred as conDetSEC (constrained DEep Time Series Embedding Clustering). Let $X = \{X_i\}_{i=1}^{n}$ be a multi-variate time-series dataset where X_i is a time-series and $X_{ij} \in R^d$ is the multi-dimensional vector of the time-series X_i at timestamp j. The maximum length of a time-series is referred as T. Given X, a set of Must-link constraints ML and Cannot-link constraints CL, the goal of conDetSEC is to partition X into a predefined number of clusters, by exploiting the supervision supplied by the set of ML and CL constraints. We remind that Must-link (resp. Cannot-link) constraints are defined over pair of time-series and indicate that two multi-variate time-series should (resp. should not) belong to the same partition [9].

The core of our framework is a neural network based architecture. We use recurrent neural network architectures [2], and more specifically a Gated Recurrent Unit (GRU) [4], to cope with both the intrinsic sequential information and with the multi-variate information that characterizes time-series data acquired by real-world sensors. A visual representation of the GRU unit is depicted in Fig. 1.

Moreover, formally speaking, a Gated Recurrent Unit (GRU) is defined as follows:

$$z_t = \sigma(W_{zx}x_t + W_{zh}h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \quad (2)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tanh(W_{hx}x_t + W_{hr}(r_t \odot h_{t-1}) + b_h) \quad (3)$$

The $\odot$ symbol indicates an element-wise multiplication while σ and tanh represent Sigmoid and Hyperbolic Tangent function, respectively. x_t is the timestamp input vector and h_{t-1} is the hidden state of the recurrent unit at time $t-1$. The different weight matrices W_{**} and bias vectors b_* are parameters learned during the training of the model.

This unit follows the general philosophy of modern Recurrent Neural Network models implementing gates and cell states. The *GRU* unit has two gates, update (z_t) and reset (r_t), and one cell state, the hidden state (h_t). Moreover, the two gates combine the current input (x_t) with the information coming from the previous timestamps (h_{t-1}). The update gate effectively controls the trade off between how much information from the previous hidden state will carry over to the current hidden state and how much information of the current timestamp needs to be kept. On the other hand, the reset gate monitors how much information of the previous

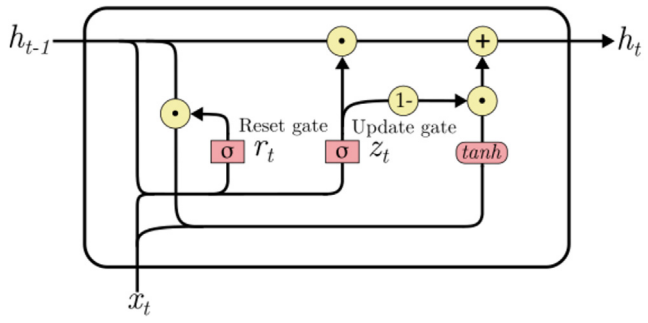

Fig. 1. Visual representation of the Gated Recurrent Unit cell. The GRU cell has two internal gates (z_t and r_t) and an hidden state (h_t). Gates are employed to combine together current information with the one coming from previous timestamps.

timestamps needs to be integrated with current information. As each hidden unit has separate reset and update gates, they are able to capture dependencies over different time scales. Units more prone to capturing short-term dependencies will tend to have a frequently activated reset gate, but those that capture longer-term dependencies will have update gates that remain mostly active [4].

The conDetSEC framework includes two main stages: embedding generation and clustering refinement. Constraints information (or semi-supervision) is integrated in both stages: during the data embedding generation, by jointly optimizing the network parameters via both unsupervised and semi-supervised tasks, and at the clustering refinement stage, where the embedding manifold is stretched towards clustering centroids. Such centroids can be derived by applying any centroid-based clustering algorithm (e.g., K-means) on the new data representation. The final clustering assignment is derived by applying the K-means clustering algorithm on the embeddings produced by conDetSEC.

Algorithm 1 summarizes the conDetSEC framework. The first stage (lines 1–7) is devoted to learn multi-variate time series representation exploiting the supervision supplied by the available constraints. To this purpose, we set up a reconstruct task in which GRU autoencoders are employed to encode/decode the sequential information. A visual summary of the autoencoder structure we have conceived is depicted in Fig. 2.

Here, we use two GRU autoencoders AE_f and AE_b. Each autoencoder has two components, an encoder (enc_f and enc_b) and a decoder (dec_f and dec_b). More precisely, AE_f encodes (enc_f) and decodes (dec_f) the original time series considering the natural order (forward) while AE_b encodes (enc_b) and decodes (dec_b) the time series in reverse order (backward) w.r.t. the time dimension. The two AE models interact with each other since, the same embedding representation, we can name it emb_i ($emb_i = enc_f(X_i) + enc_b(rev(X_i))$), is fed to the two decoders. Here, X_i is the time-series i and $rev(X_i)$ is the same time-series in reverse order w.r.t. the time dimension. smb_i refers to the embedding for the time-series X_i. Furthermore, Θ_1, Θ_2 and Θ_3 are the learnt parameters of the recurrent model where Θ_1 are the parameters associated to the two encoders (enc_f and enc_b) and Θ_2 (resp. Θ_3) are the parameters associated to the forward (resp. backward) decoders dec_f (resp. dec_b). The reconstruction error is assessed by means of standard Mean Squared Error loss (or squared L2 norm) between the original time-series and the reconstructed one (line 4).

Furthermore, after that the model parameters are modified by means of the two (forward and backward) reconstruction loss functions, the semi-supervision is integrated through the L_{Contr} loss that has the objective to take into account the information carried out by the set of *ML* and *CL* constraints. To this end, we leverage a contrastive loss [5] with the aim to inject the available knowledge in the representation learnt by the main reconstruction process. More formally, we can define the contrastive loss, in our context, as follows:

Algorithm 1: conDetSEC Optimization

Require: X, ML, CL, N_PRET_EPOCHS, N_REFINE_EPOCHS, $nClust$.
Ensure: *embeddings*.
1: i = 0
2: **while** i $<$ N_PRET_EPOCHS **do**
3: Update Θ_1, Θ_2 and Θ_3 by descending the gradient:
4: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X|}\sum_{i=1}^{|X|}||X_i - AE_f(X_i|\Theta_1,\Theta_2)||_2^2 + \frac{1}{|X|}\sum_{x_i\in X}||rev(X_i) - AE_{back}(X_i|\Theta_1,\Theta_3)||_2^2$ with mini-batch SGD
5: $\nabla_{\Theta_1} \frac{1}{|ML\cup CL|} L_{Contr}(ML, CL|\Theta_1)$ with mini-batch SGD
6: i = i + 1
7: **end while**
8: *embeddings* = extractEmbedding(Θ_1, X)
9: δ, C = runKMeans(embeddings, nClust)
10: i = 0
11: **while** $i < N\_REFINE\_EPOCHS$ **do**
12: Update Θ_1, Θ_2 and Θ_3 by descending the gradient:
13: $\nabla_{\Theta_1,\Theta_2,\Theta_3} \frac{1}{|X|}\sum_{i=1}^{|X|}||X_i - AE_f(X_i|\Theta_1,\Theta_2)||_2^2 + \frac{1}{|X|}\sum_{x_i\in X}||rev(X_i) - AE_{back}(X_i|\Theta_1,\Theta_3)||_2^2 +$
$L_{stretch}(embeddings, \delta, C|\Theta_1)$ with mini-batch SGD
14: $\nabla_{\Theta_1} \frac{1}{|ML\cup CL|} L_{Contr}(ML, CL|\Theta_1)$ with mini-batch SGD
15: i = i + 1
16: **end while**
17: *embeddings* = extractEmbedding(Θ_1, X)
18: **return** *embeddings*

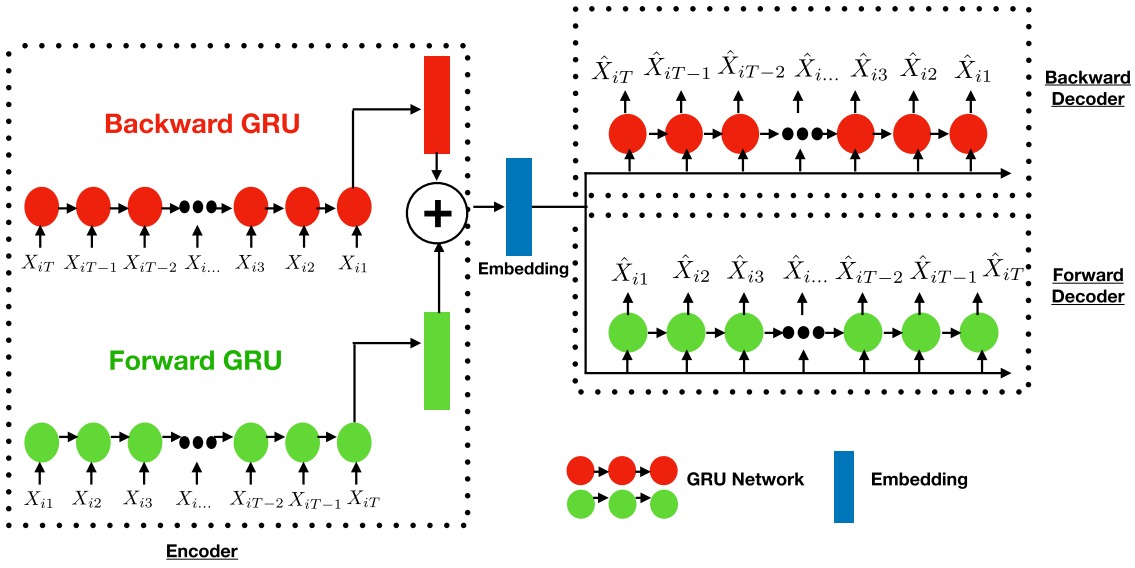

Fig. 2. Encoder/Decoder structure of conDetSEC. The network has three main components: i) an encoder, ii) a forward decoder and iii) a backward decoder. The encoder includes forward/backward GRU networks. From each GRU encoder an embedding is extracted. Subsequently, the per-encoder embedding are combined via a point-wise sum operation to obtain the final embedded representation. Finally, the forward decoder reconstructs the original signal considering its original order (forward - green color) while the backward decoder reconstructs the same signal but in inverse order (backward - red color). $\hat{X}_{i..}$ indicates reconstructed information per timestamps.

$$L_{Contr} = \sum_{(X_i,X_j)\in ML} \frac{1}{2} Dist(X_i,X_j) + \sum_{(X_i,X_j)\in CL} \frac{1}{2}[m - Dist(X_i,X_j)]_+ \quad (4)$$

where $Dist(X_i,X_j)$ is the squared euclidean distance between the embeddings of the time-series X_i and X_j ($Dist(X_i,X_j) = ||emb_i - emb_j||_2^2$), $[c]_+$ is the classical hinge loss defined as max(0,c) and m is the margin hyperparameter. The idea behind this loss is to minimize the distance between two embedded multi-variate time-series that should belong (ML) to the same partition (first term of the equation) as well as maximize the distance (up to a certain margin) between two embedded multi-variate time-series that should not belong (CL) to the same partition (second term of the equation). We underline that, at each epoch, the procedure firstly iterates on the whole set of unlabelled time-series data X updating parameters Θ_1, Θ_2 and Θ_3 (line 4) and, successively, it iterates through the set of constraints ($ML \cup CL$) updating the network parameters Θ_1 (line 5).

Then, the pretrained encoders are employed to extract the multi-variate time-series representation with the aim to compute an initial clustering of the data from which centroids are derived (lines 8–9).

Successively (lines 11–15), the clustering solution is refined by means of an additional term in the loss function defined as follows:

$$L_{stretch}(embeddings, \delta, C|\Theta_1) = \frac{1}{|X|} \sum_{emb_i \in embeddings} \sum_{j=1}^{nClust} \delta_{ij} ||C_j - emb_i||_2^2 \quad (5)$$

where $nClust$ is the number of cluster, $\delta_{ij}$is an indicator function that expresses if a time-series i belongs to the cluster j and C_j is the centroid of cluster j.

Such a term allows to explicitly stretch the manifold embedding with the aim to move closer the time-series embeddings with the corresponding clustering centroids inducing a more sharp clustering structure. Similarly to what was done for the first stage of the procedure, also for this part of the framework, at each epoch, firstly the reconstruction as well as the stretching loss terms are optimized considering the whole set of unlabelled time-series data X modifying the parameters Θ_1, Θ_2 and Θ_3 (line 13) and, only successively, the contrastive loss is optimized on the set of constraints ($ML \cup CL$) updating the network parameters Θ_1 (line 14).

More generally, the second stage of conDetSEC is devoted to further stretch the manifold on which the embeddings lie moving them closer to their corresponding centroids and, simultaneously, still pay attention to the background knowledge provided under the shape of ML and CL constraints. Moving the embeddings closer to their corresponding centroids helps to achieve a more sharp structure avoiding possible confusion in the clustering assignment related to the non-deterministic nature of the majority of modern clustering algorithms. All these components modify the new data representation (embeddings) with the goal to facilitate the work of the downstream algorithm that will be employed to provide the final result.

Finally, the new data representation (*embeddings*) is extracted (line 15) and returned by the procedure. The final partition is obtained by applying the K-Means clustering algorithm on the new data representation.

4. Experiments

In this section we describe the experimental evaluation we have conducted to assess the performance of conDetSEC. To this end, we compare conDetSEC with different competing methods over several benchmarks and we quantitatively and qualitative inspect the behaviour of our proposal.

4.1. Competitors

For the comparative study, we consider the following competitors:

- The well-established *K-means* clustering algorithm [27] equipped with Dynamic Time Warping distance measure [8] (*DTW*).
- Another version of *K-means* algorithm equipped with the Soft Dynamic Time Warping measures introduced in [7] (*SOFTDTW*). This measure is a differentiable distance measure recently introduced to manage dissimilarity evaluation between multivariate time-series of variable length;

- The constrained spectral clustering approach proposed in [19], named *Spec*. Such a clustering algorithm is still based on DTW as internal dissimilarity measure. Among several methods for constrained clustering over time series data, the spectral based algorithm exhibits the best performances;
- The COP *K-means* algorithm coupled with DTW. In this case, the original k-Means algorithm is modified in order to choose a reassignment of the clustering solution not violating any constraints at each iteration. Similarly to the spectral based approach, also this method is introduced in [19] and it is pointed out as one of the strategies achieving the highest clustering performances when a set of diverse time series benchmarks is considered. We name such a competitor COPK-Means.
- A fully unsupervised variant of our approach that does not consider the input constraints. This baseline allows to analyze the added value related to the injection of background knowledge under the form of constraints. We name such a baseline conDetSEC$_{noC}$.

All the k-means based approaches are adapted to multi-variate time-series analysis leveraging the DBA (DTW Barycenter Averaging) method [20].

Summing up, the two competitors based on the original *K-means* algorithm (*DTW* and *SOFTDTW*), as well as the conDetSEC$_{noC}$ variant of the proposed approach, are fully unsupervised, i.e., they do not take into account the ML and CL constraints. While the constrained spectral clustering approach (*Spec*) and the COP *K-means* algorithm are able to exploit constraints, they do so by adapting classic constrained clustering algorithms to the time series domain, i.e., constraints are exploited through basic algorithmic rules and distance measures. Conversely, the aim of conDetSEC is to exploit an advanced neural network architecture in order to overcome the previous strategies, i.e., by explicitly taking into account the constraints while at the same time managing the time dimension during the process to learn the new embedded representation.

4.2. Data and Experimental Settings

The evaluation has been carried out on six benchmarks [11] coming from disparate application domains and characterized by contrasted features in terms of number of samples, number of attributes (dimensions) and time series lengths. More in detail:

- The *ArabDigit* dataset contains timeseries of mel-frequency cepstrum coefficients (MFCCs) corresponding to spoken Arabic digits. It includes data from 44 male and 44 female native Arabic speakers.
- The *JapVowel* benchmark still involves a speech classification task. Nine male speakers uttered two Japanese vowels/ae/ successively. For each utterance discrete-time series of LPC cepstrum coefficients are extracted.
- The *Dordogne* benchmark involves a classification task where time series are extracted from multi-temporal satellite image acquisitions. The underlying task consists in the classification of each (pixel) multi-variate time series to one of the several land covers available in the nomenclature associated to the study site (i.e. Crop, Water, Urban settlement, Forest, etc...).
- The *ECG*5000 benchmark contains time series coming from the medical domain and describing people with severe congestive heart failure. The class values were obtained by automated annotation. The data was pre-processed in two steps: (1) extract each heartbeat, (2) make each heartbeat equal length using interpolation.
- The *HAR* (Human Activity Recognition) dataset has been collected from 30 subjects performing six different activities (Walking, Walking Upstairs, Walking Downstairs, Sitting, Standing, Laying). It consists of inertial sensor data that was collected using a smartphone carried by the subjects.
- The *Pendigits* benchmark involves a handwritten digit classification task. 44 writers were asked to draw the digits [0...9], where instances are made up of the x and y coordinates of the pen traced across a digital screen. The data was spatially resampled to 8 spatial points, such that each sample has 2 dimension of 8 points, with a single class label [0...9] being the digit drawn.

All benchmarks, except *Dordogne* (which was obtained by contacting the authors of [13]), are available online.

To measure the performances of all the clustering methods, we use the Normalized Mutual Information (NMI) [25] as well as the Adjusted Rand Index (ARI) [15]. Both metrics take their maximum value when the clustering partition completely matches the original one, i.e., the partition induced by the available class labels. The NMI measure ranges between $[0, 1]$ while the ARI index ranges between $[-1, 1]$. Both evaluation metrics can be considered as an indicator of the purity of the clustering result.

We analyze the behavior of the different methods according to increasing levels of supervision. We simulate the supervision by randomly select Must-link and Cannot-link constraints similarly to what is commonly done in previous studies on semi-supervised clustering [34,1,23]. More precisely, we randomly sample pairs of points and for each pair, we introduce a Cannot- or Must-link constraint based on the labels of the sampled pair. We adopt a similar constraint range as the one proposed in [38], we vary the total amount of constraints from 1 000 to 5 000 with a step of 1 000. Such a constraint range represents a very small portion of all the possible constraints we can generate from the previously presented benchmarks. In addition, we have empirically observed that considering a smaller number of constraints does not introduce any kind of supervision in the competing approaches. For this reason, we have chosen a constraint range that can provide room for investigation according with the considered competing methods. Due to the random sample selection process and the non deterministic nature of the clustering algorithms, we repeat the sample selection step 5 times for each number of constraints. Finally, for each level of supervision, we report the average values of Normalized Mutual Information and Adjusted Rand Index. For all the methods, the number of clusters is equal to the number of classes.

conDetSEC is implemented via the *Tensorflow* python library and the implementation is available online [1]. Model parameters are learnt using the Adam optimizer [18] with a learning rate equal to 5×10^{-4} for both stages of our framework (embedding generation and clustering refinement).

We set a batch size of 32 and the number of pretraining (*N_PRET_EPOCHS*) and refining (*N_REFINE_EPOCHS*) epochs to 40 and 60, respectively. We set to 64 the number of hidden units for the GRU cell. We remind that, for our framework, conDetSEC, exactly the same architecture is employed for all the different benchmarks involved in the experimental evaluation.

For all the competing methods we use the TSLEARN python library [28] implementation of the Dynamic Time Warping and Soft Dynamic Time Warping measure. Experiments are carried out on a workstation equipped with an Intel(R) Xeon(R) W-2133, 3.6Ghz CPU, with 64 Gb of RAM and one GTX1080 Ti GPU.

[1] https://gitlab.irstea.fr/dino.ienco/cmts-clustering/

4.3. Results

Fig. 3 and Fig. 4 summarize the results, in terms of average Normalized Mutual Information and Adjusted Rand Index, respectively, of the different competing methods over the multi-variate time series benchmarks varying the amount of labelled samples from which constraints are derived. Generally, we can observe that both evaluation metrics, NMI and ARI, depict a similar scenario. conDetSEC systematically outperforms all the competing approaches when at least two thousand constraints are taken into account. Note that conDetSEC generally outperforms competing methods also for the one thousand constraints configuration, with some rare exceptions (*ECG*5000 for ARI, *HAR* for both metrics). In addition, we can also note that our framework tends to provide better clustering solutions when the amount of available labelled samples per class increases. Conversely, the other semi-supervised clustering methods (*Spec* and COP-KMeans) clearly struggle to take advantage of increasing amount of background

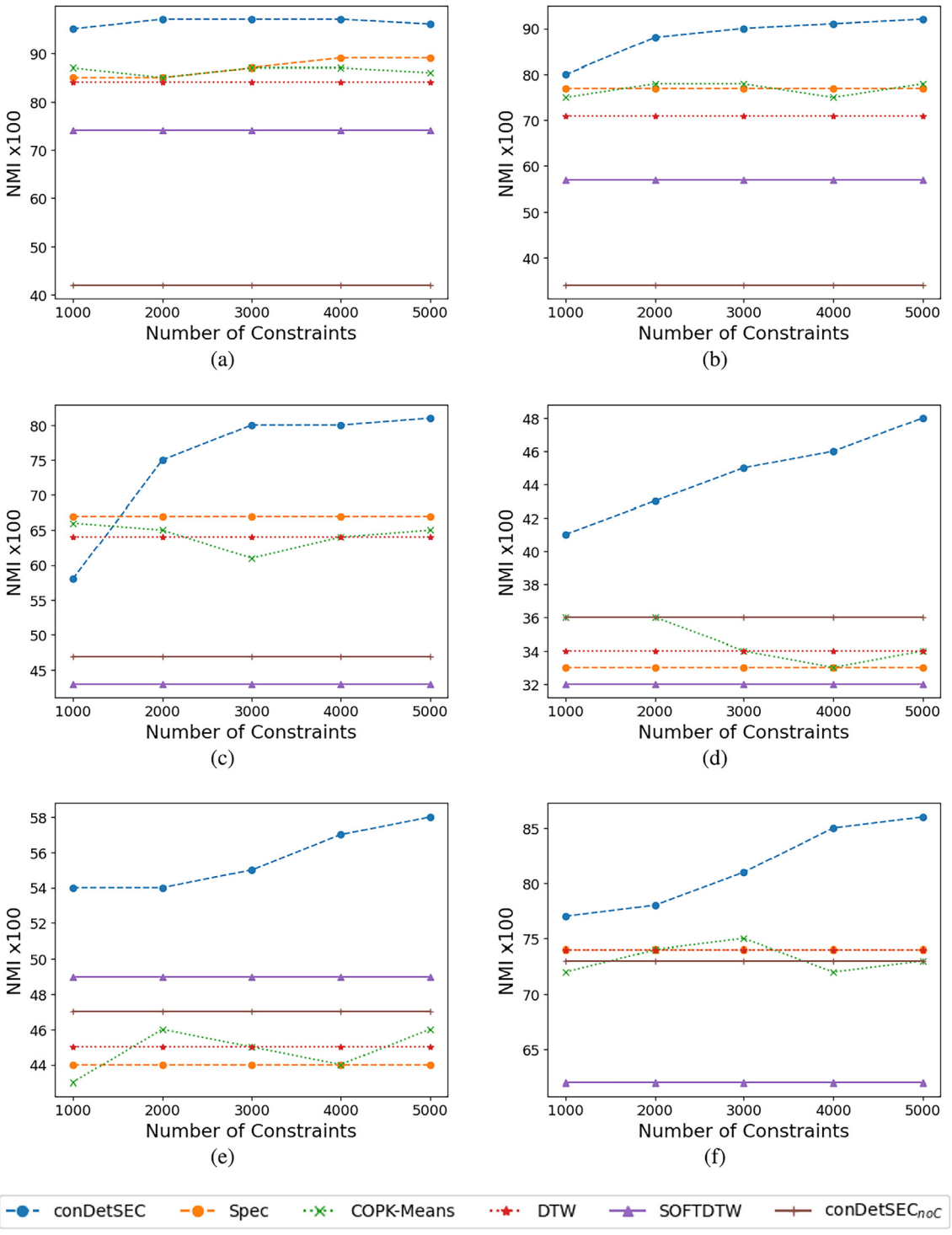

Fig. 3. Results (in terms of NMI) of the different approaches varying the amount of constraints on: (a) JapVowel (b) ArabDigit (c) HAR (d) Dordogne (e) ECG5000 and (f) PenDigits benchmarks.

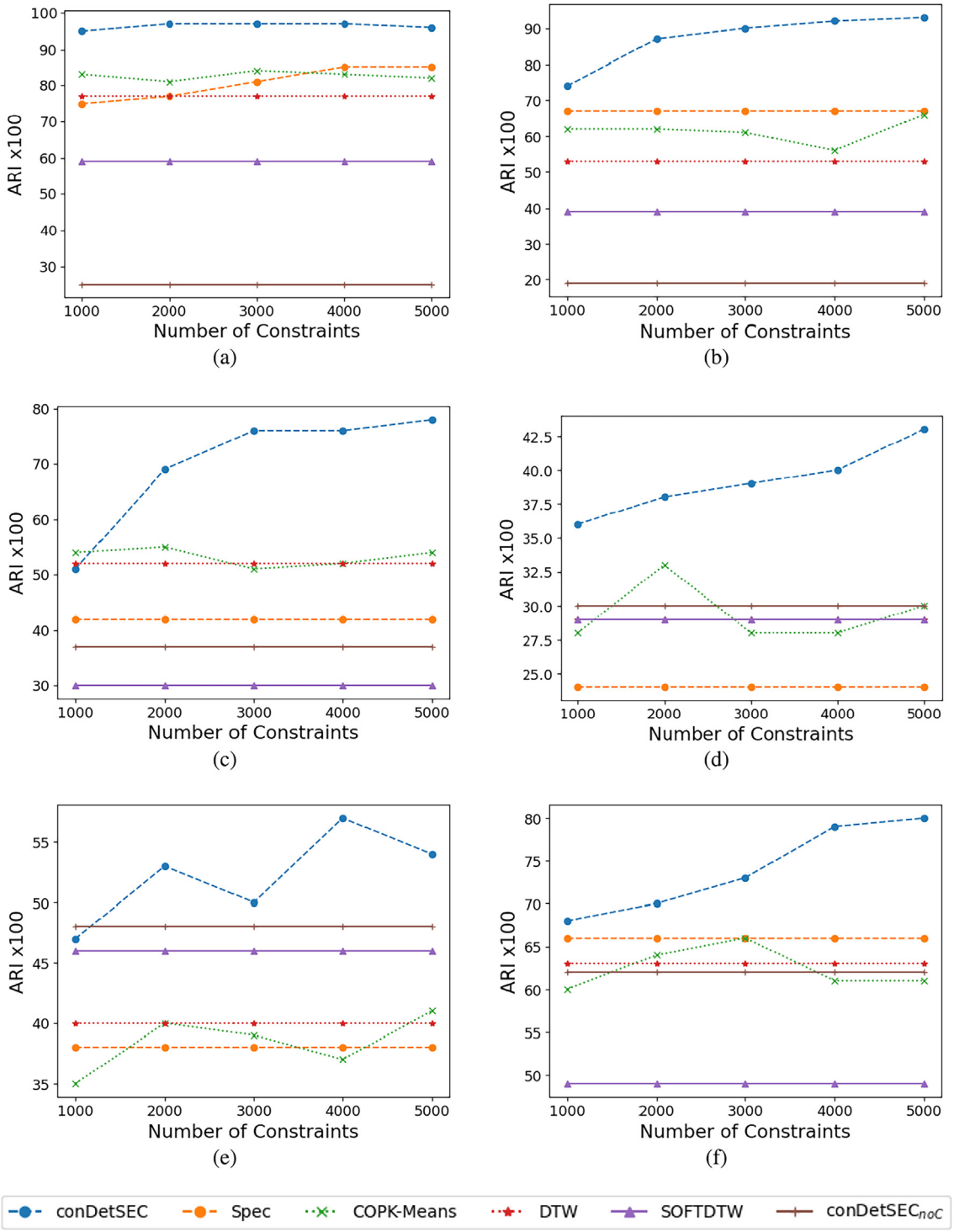

Fig. 4. Results (in terms of ARI) of the different approaches varying the amount constraints on: (a) JapVowel (b) ArabDigit (c) HAR (d) Dordogne (e) ECG5000 and (f) PenDigits benchmarks.

knowledge to steer the learning process. *Spec* generally exhibits stable performance metric values, no matter the amount of background knowledge is injected in the semi-supervised process, with the only exception being *JapVowel* (where its performance slightly improves with the number of constraints). Conversely, COP-KMeans shows to be sensitive to the number of constraints taken into account, but rarely with positive outcomes (i.e., its performance tends to decrease when a higher number of constraints is exploited).

Regarding the fully unsupervised approaches *DTW* and *SOFTDTW*, we can notice that the former outperforms the latter in the majority of the cases. In addition, the K-means strategy based on the *DTW* measure exhibits competitive performances with respect to its semi-supervised counterparts.

A direct comparison between conDetSEC and *DTW* also indicates that *HAR* is the only benchmark over six where, when the smallest amount of constraints is taken into account (one thousand), the fully unsupervised *DTW* clustering approach exhibits

competitive performances with respect to conDetSEC. Then, when reasonable amount of background knowledge is integrated in the learning process, our framework clearly exploits such information to boost the clustering performances achieving evident gains compared to the *DTW* method. Finally, the comparison between conDetSEC and its fully unsupervised ablation (conDetSEC$_{noC}$) makes evident the fact that the proposed framework effectively exploits the amount of supervision it can access.

Fig. 5 and Fig. 6 depict the performances in terms of Normalized Mutual Information and Adjusted Rand Index, respectively, when only the samples involved in the Must-link and Cannot-link constraints are considered. This experiment has the objective to investigate how the different competing methods internally manage the samples related to the background knowledge, thus providing room for understanding the ability of each competing approach to satisfy the set of input constraints. Since the COP *K-means*

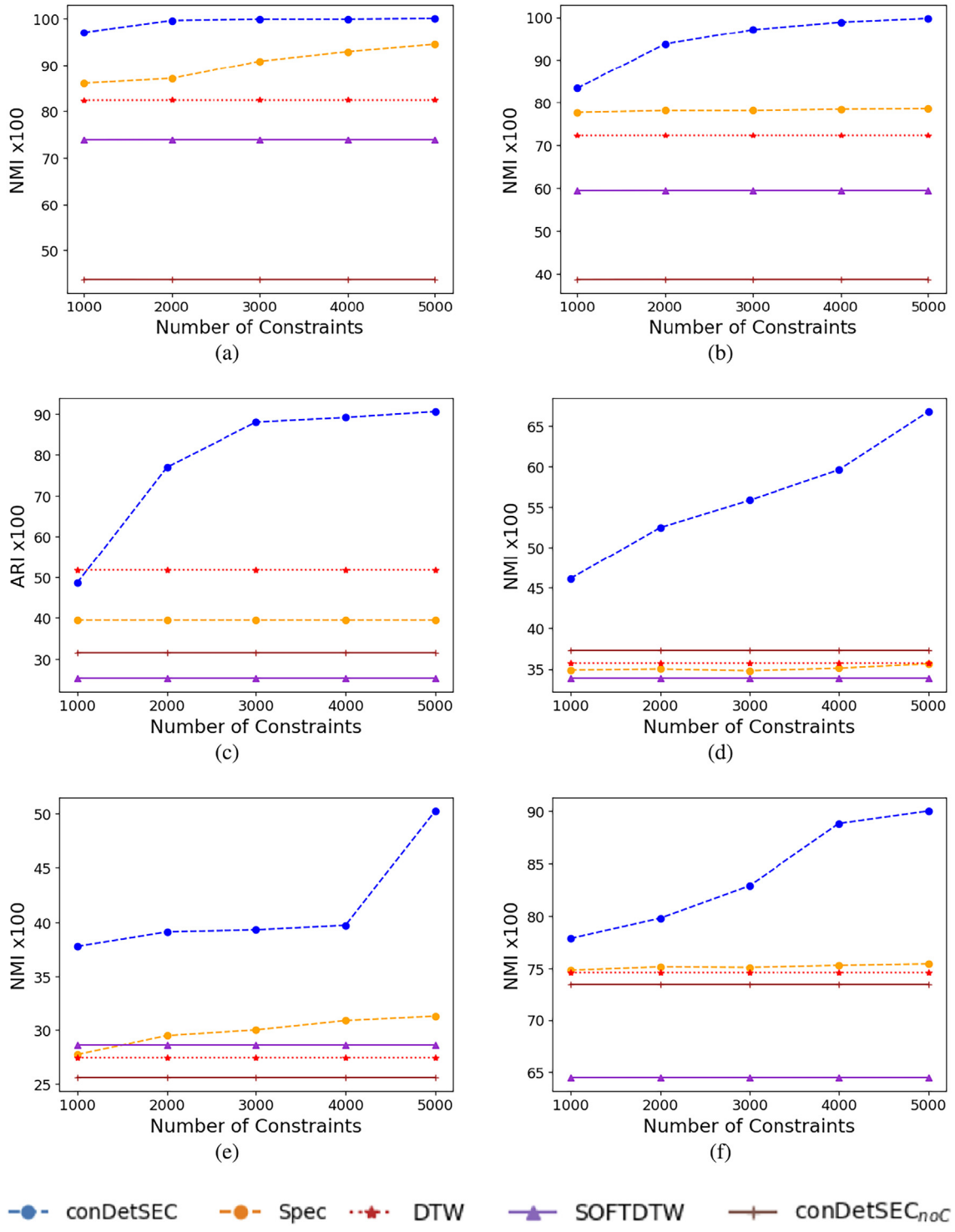

Fig. 5. Results (in terms of NMI) considering only the samples involved in the set of Must-link and Cannot-link constraints on: (a) JapVowel (b) ArabDigit (c) HAR (d) Dordogne (e) ECG5000 and (f) PenDigits benchmarks.

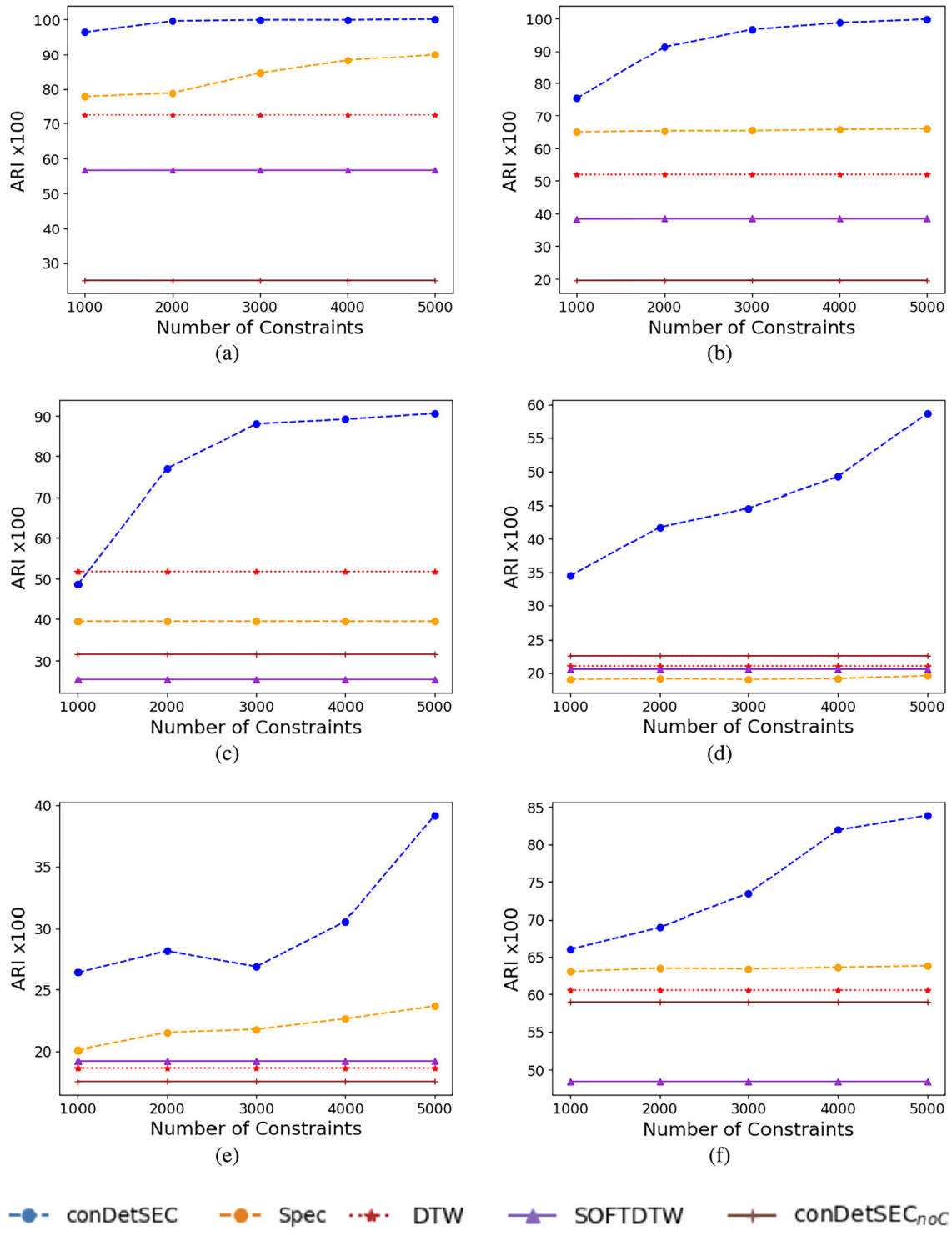

Fig. 6. Results (in terms of ARI) considering only the samples involved in the set of Must-link and Cannot-link constraints on: (a) JapVowel (b) ArabDigit (c) HAR (d) Dordogne (e) ECG5000 and (f) PenDigits benchmarks.

method satisfies the whole set of input constraints by construction, we did not include it for this experiment.

We can note that both NMI and ARI measures provide a similar picture of the results. conDetSEC exhibits the best performances over all the considered set of benchmarks under all constraints configurations (the only exception being *HAR* with 1000 constraints), demonstrating once again its strong ability to well capture the supervision supplied by the set of input constraints conversely to the other competitors. On some benchmarks (i.e., *JapVowel* and *ArabDigit*), our framework achieves a perfect matching when enough input constraints are provided as input to the semi-supervised time series clustering process. While for other benchmarks (e.g., *HAR* and *PenDigits*) the performance of conDetSEC, in terms of absolute values, is also excellent (NMI and ARI around 0.9), for *Dordogne* and *ECG*5000 we can note slightly lower behaviors. This attitude is shared by all the competing approaches,

Table 1
Benchmark Characteristics.

Dataset	# Samples	# Dims	Min/Max Length	Avg. Length	# Classes
ArabDigit	8800	13	4/93	39	10
JapVowel	640	12	7/29	15	9
Dordogne	9919	6	23/23	23	7
ECG5000	4686	1	140/140	140	5
HAR	10299	9	128/128	128	6
PenDigits	10992	2	8/8	8	10

pinpointing the fact that these datasets, compared to the other benchmarks employed in the experimental evaluation, constitute a challenging testbed for the semi-supervised learning algorithms. Finally, we can also observe that conDetSEC clearly benefits from the increasing amount of Must- and Cannot-link constraints, conversely to the competing approaches that are marginally impacted by the amount of available background knowledge. Table 1.

Table 2 reports an additional qualitative experiment where the distance distributions for the samples involved in the Must-link and Cannot-link constraints are evaluated. More in detail, here, we compute the average intra pairwise distance for the set of Must-link and Cannot-link constraints for: i) the Dynamic Time

Table 2
Statistics (average and standard deviation) about the distances between samples involved in the Must-link and Cannot-link constraint sets when Dynamic Time Warping is applied on the original time-series space (DTW) as well as the euclidean distance is deployed on the embedded space generated by conDetSEC when 5000 Must-link and Cannot-link constraints are randomly sampled.

Dataset	DTW		Embedding	
	ML	CL	ML	CL
JapWov	0.1322 ± 0.0781	0.2583 ± 0.0882	0.0991 ± 0.0705	0.5909 ± 0.1436
ArabDigit	0.1723 ± 0.0269	0.2058 ± 0.0221	0.1295 ± 0.0665	0.5067 ± 0.1116
HAR	0.2602 ± 0.1446	0.4372 ± 0.1734	0.0703 ± 0.0758	0.5132 ± 0.2487
Dordogne	0.2470 ± 0.0903	0.3093 ± 0.1030	0.2519 ± 0.1211	0.4257 ± 0.1535
ECG5000	0.2583 ± 0.1789	0.3557 ± 0.1562	0.1852 ± 0.1211	0.4410 ± 0.2097
PenDigits	0.3133 ± 0.1594	0.5067 ± 0.1217	0.1602 ± 0.1263	0.5450 ± 0.1462

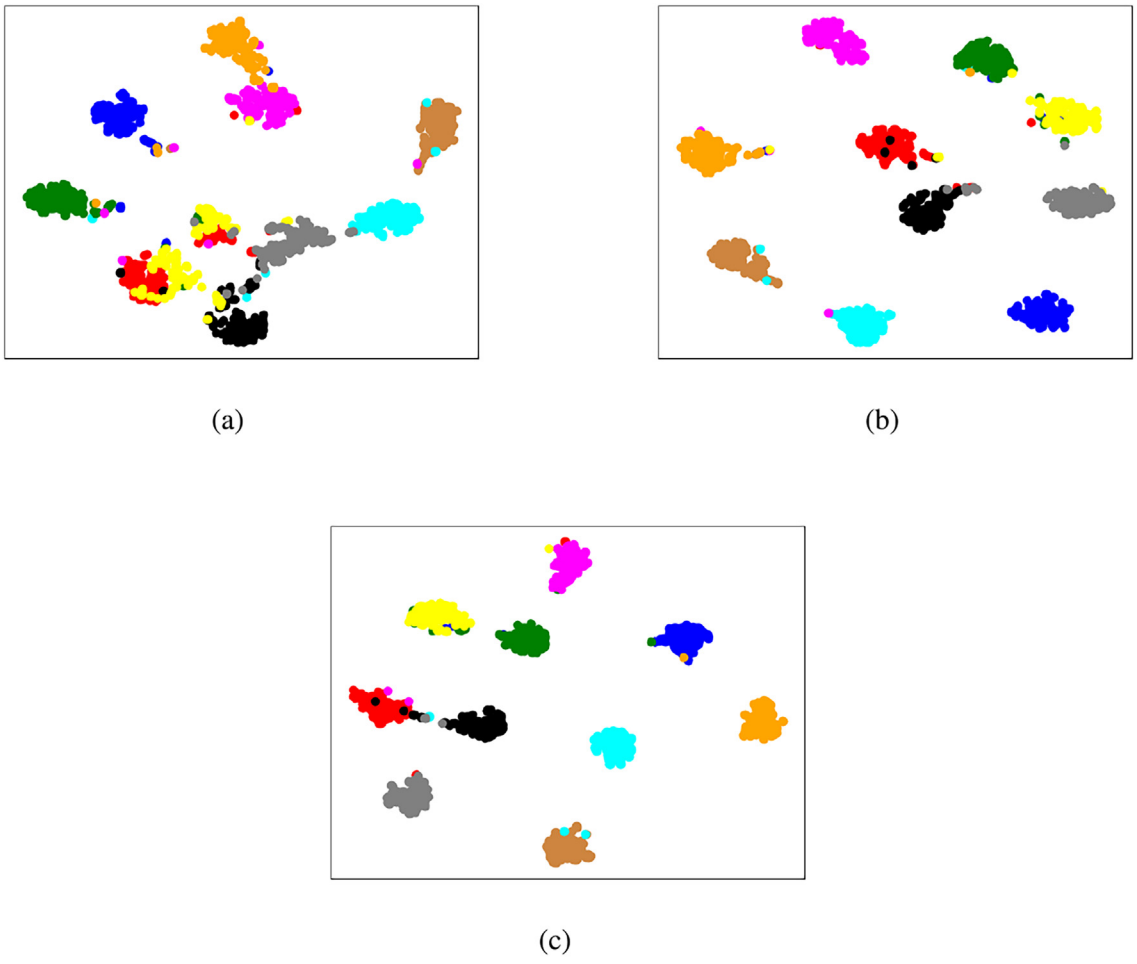

Fig. 7. Visualizaton of the conDetSEC embeddings considering the same 100 per class samples belonging to the *ArabDigits* dataset with (a) 1000 (b) 3000 and (c) 5000 randomly sampled constraints. The two dimensional representation is obtained via the T-SNE algorithm [33].

Warping distance applied on the original time-series space (DTW) and ii) the euclidean distance on the embedded space generated by conDetSEC (Embedding). For each pair of (dataset,method) we report average and standard deviation. Results have been obtained by exploiting 5000 random sampled constraints. We can note that, for all benchmarks, the embedded space generated by conDetSEC permits to clearly stretch the manifold facilitating a better separation (in terms of average distance) between the samples involved in the Cannot-link constraints with respect to the samples involved in the Must-link ones.

To conclude the experimental evaluation, with the aim to visually inspect the behavior of conDetSEC, we depict the embedding generated by our approach on the *ArabDigit* dataset by varying the amount of random sampled constraints with values $1000, 3000$ and 5000. The results of this analysis are reported in Fig. 7. To obtain the two dimensional representations, we apply the *t*-distributed stochastic neighbor embedding (*TSNE*) approach [33]. For this evaluation we consider 100 instances per class. To make the visual results comparable, we firstly group together all the embedding generated by conDetSEC, no matter the amount of class samples were used to generated them. Successively, we deploy the *TSNE* approach and, finally, we separated the obtained projections to retrieve the original partitions. In this way, all the *TSNE* projections exist in the same two dimensional space making the visual inspection fair. The experiment underlines the ability of conDetSEC to modify the data manifold exploiting the increasing amount of background knowledge. We observe that clear differences exist between the embeddings learnt when 1000 (Fig. 7 (a)), 3000 (Fig. 7(b)) and 5000 (Fig. 7(c)) randomly sampled constraints are considered. Generally, increasing the amount of semi-supervision results in more clear cluster structure in which the classes are more distinguishable and separated from each other. As a final remark, we want to stress out how conDetSEC proved to be effective (constantly outperforming competing approaches) on benchmarks coming from different application domains and characterized by different structural characteristics. This proves how the proposed approach is extremely flexible, since it has not been designed to work on a specific domain or on time series of a specific form. Therefore, it can be successfully applied to any real world application where it is possible to model the input data in the form of a time series with associated ML/CL knowledge (e.g., any problem relying on sensor based data and having some ground truth knowledge associated to derive the constraints set).

5. Conclusion

In this work we have presented conDetSEC, a new semi-supervised (constrained) clustering algorithm especially tailored for multi-variate time series data. The proposed framework is based on Gated Recurrent Unit models and it includes two different stages: embedding generation and clustering refinement. Constraints information (or semi-supervision) is integrated in both stages.

The evaluation on six benchmarks has demonstrated the effectiveness of conDetSEC and its flexibility on data coming from different application domains. The achieved results clearly point out that conDetSEC has the ability to effectively exploits the amount of supervision it can access. Additionally, we also conduct a visual inspection of the embedded representation learnt by conDetSEC that shows how the manifold learnt by the learning process is positively influenced by the amount of available background knowledge injected as pairwise constraints. Despite the interesting behavior exhibited by conDetSEC, several limitations are still affecting our framework and need further research efforts. In the conducted research, we have made the strong assumption that clustering is performed once time series were collected. In a more realistic scenario multi-variate time series data could be acquired in a dynamic scenario (i.e. IoT or mobile sensors) where streams of information are generated. In such a context, incremental learning constitutes a possible way to limit human intervention and meet the specificity of data streams like concept drift as well as restricted access to the incoming data. Another limitation associated to our framework is that, as of now, conDetSEC is not capable to manage new must- or cannot-link constraints that can be provided in a successive moment. To this end, methods and or mechanisms related to continual or incremental learning can represent an interesting research track to explore in the future in order to cope with current limitations associated to our methodology.

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.

Acknowledgment

This work was supported by the French National Research Agency in the framework of Herelles project (ANR Project-20-CE23-0022).

References

[1] S. Basu, M. Bilenko, R.J. Mooney, A probabilistic framework for semi-supervised clustering, in: KDD, ACM, 2004, pp. 59–68.
[2] Y. Bengio, A.C. Courville, P. Vincent, Representation learning: A review and new perspectives, IEEE TPAMI 35 (8) (2013) 1798–1828.
[3] S. Chandrakala and C. Chandra Sekhar. A density based method for multivariate time series clustering in kernel feature space. In IJCNN, pages 1885–1890, 2008.
[4] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In EMNLP, pages 1724–1734, 2014.
[5] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In CVPR, pages 539–546, 2005.
[6] R. Coppi, P. D'Urso, P. Giordani, A fuzzy clustering model for multivariate spatial time series, J. Classification 27 (1) (2010) 54–88.
[7] M. Cuturi and M. Blondel. Soft-dtw: a differentiable loss function for time-series. In ICML, pages 894–903, 2017.
[8] H. Anh Dau, D. Furtado Silva, F. Petitjean, G. Forestier, A.J. Bagnall, A. Mueen, E. J. Keogh, Optimizing dynamic time warping's window width for time series data mining applications, Data Min. Knowl. Discov. 32 (4) (2018) 1074–1120.
[9] I. Davidson and S.S. Ravi. Intractability and clustering with constraints. In ICML, pages 201–208, 2007.
[10] P. D'Urso, E. Ann Maharaj, Wavelets-based clustering of multivariate time series, Fuzzy Sets and Systems 193 (2012) 33–61.
[11] 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 (4) (2019) 917–963.
[12] C. Hora Fontes, I. Celestina Santos, M. Embiruçu, P. Aragão, Pattern reconciliation: A new approach involving constrained clustering of time series, Comput. Chem. Eng. 145 (2021) .
[13] Y. Jean Eudes Gbodjo, D. Ienco, and L. Leroux. Toward spatio-spectral analysis of sentinel-2 time series data for land cover mapping. IEEE GRSL, 17(2):307–311, 2020.
[14] D. Hallac, S. Vare, S.P. Boyd, and J. Leskovec. Toeplitz inverse covariance-based clustering of multivariate time series data. In KDD, pages 215–223, 2017.
[15] L. Hubert, P. Arabie, Comparing partitions, Journal of Classification 2 (1) (1985) 193–218.
[16] D. Ienco and R. Interdonato. Deep multivariate time series embedding clustering via attentive-gated autoencoder. In PAKDD, pages 318–329, 2020.
[17] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate lstm-fcns for time series classification, Neural Networks 116 (2019) 237–245.
[18] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, CoRR abs/1412.6980 (2014).
[19] T.A. Lampert, T.-B.-H. Dao, B. Lafabregue, N. Serrette, G. Forestier, B. Crémilleux, C. Vrain, P. Gançarski, Constrained distance based clustering for time-series: a comparative and experimental study, Data Min. Knowl. Discov. 32 (6) (2018) 1663–1707.

[20] T.A. Lampert, B. Lafabregue, T.-B.-H. Dao, N. Serrette, C. Vrain, P. Gançarski, Constrained distance-based clustering for satellite image time-series, IEEE J. Sel. Top. Appl. Earth Obs. Remote. Sens. 12 (11) (2019) 4606–4621.
[21] T. Warren Liao, Clustering of time series data - a survey, Pattern Recognition 38 (11) (2005) 1857–1874.
[22] F. Liu, M. Cai, L. Wang, Y. Lu, An ensemble model based on adaptive noise reducer and over-fitting prevention LSTM for multivariate time series forecasting, IEEE Access 7 (2019) 26102–26115.
[23] S. Sundar Rangapuram and M. Hein. Constrained 1-spectral clustering. In AISTATS, volume 22 of JMLR Proceedings, pages 1143–1151, 2012.
[24] S.-Y. Shih, F.-K. Sun, H.-Y. Lee, Temporal pattern attention for multivariate time series forecasting, Machine Learning 108 (8–9) (2019) 1421–1441.
[25] A. Strehl, J. Ghosh, Cluster ensembles — A knowledge reuse framework for combining multiple partitions, Journal of Machine Learning Research 3 (2002) 583–617.
[26] R.L. Talavera-Llames, R. Pérez-Chacón, A. Troncoso, F. Martínez-Álvarez, Mv-kwnn: A novel multivariate and multi-output weighted nearest neighbours algorithm for big data time series forecasting, Neurocomputing 353 (2019) 56–73.
[27] P.-N. Tan, M. Steinbach, V. Kumar, Introduction to Data Mining, First Edition., Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA, 2005.
[28] R. Tavenard, J. Faouzi, G. Vandewiele, F. Divo, G. Androz, C. Holtz, M. Payne, R. Yurchak, M. Rußwurm, K. Kolar, E. Woods, Tslearn, a machine learning toolkit for time series data, Journal of Machine Learning Research 21 (118) (2020) 1–6.
[29] D.J. Trosten, A. Storvik Strauman, M. Kampffmeyer, R. Jenssen, Recurrent deep divergence-based clustering for simultaneous feature learning and clustering of variable length time series, in: ICASSP, IEEE, 2019, pp. 3257–3261.
[30] P. Tzirakis, M.A. Nicolaou, B.W. Schuller, and S. Zafeiriou. Time-series clustering with jointly learning deep representations, clusters and temporal boundaries. In ICAFGR, pages 1–5, 2019.
[31] T. van Craenendonck, S. Dumancic, E. Van Wolputte, and H. Blockeel. COBRAS: interactive clustering with pairwise queries. In IDA, pages 353–366, 2018.
[32] T. van Craenendonck, W. Meert, S. Dumancic, and H. Blockeel. COBRASTS: A new approach to semi-supervised clustering of time series. In Discovery Science, pages 179–193, 2018.
[33] L. van der Maaten, G. Hinton, Visualizing Data Using t-SNE, Journal of Machine Learning Research 9 (2008) 2579–2605.
[34] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, Constrained k-means clustering with background knowledge, in: ICML, Morgan Kaufmann, 2001, pp. 577–584.
[35] E.H.C. Wu and P.L.H. Yu. Independent component analysis for clustering multivariate time series data. In ADMA, pages 474–482, 2005.
[36] G. Wu, H. Zhang, Y. He, X. Bao, L. Li, X. Hu, Learning kullback-leibler divergence-based gaussian model for multivariate time series classification, IEEE Access 7 (2019) 139580–139591.
[37] Z. Xie, H. Hu, Q. Wang, R. Li, Algenet: Adaptive log-euclidean gaussian embedding network for time series forecasting, Neurocomputing 423 (2021) 353–361.
[38] H. Zhang, T. Zhan, S. Basu, I. Davidson, A framework for deep constrained clustering, Data Min. Knowl. Discov. 35 (2) (2021) 593–620.
[39] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu. Tapnet: Multivariate time series classification with attentional prototypical network. In AAAI, pages 6845–6852, 2020.

Dino Ienco received the M.Sc. and Ph.D. degrees in computer science both from the University of Torino, Torino, Italy, in 2006 and 2010, respectively. He joined the TETIS Laboratory, IRSTEA, Montpellier, France, in 2011 as a Junior Researcher. His main research interests include machine learning, data science, graph databases, social media analysis, information retrieval and spatio-temporal data analysis with a particular emphasis on remote sensing data and Earth Observation data fusion. Dr. Ienco served in the program committee of many international conferences on data mining, machine learning, and database including IEEE ICDM, ECML PKDD, ACML, IJCAI as well as served as a Reviewer for many international journal in the general field of data science and remote sensing.

Roberto Interdonato is a Research Scientist at Cirad, UMR TETIS, Montpellier, France. He was previously a post-doc researcher at University of La Rochelle (France), Uppsala University (Sweden) and at University of Calabria (Italy), where he received his Ph.D. in computer engineering in 2015. His Ph.D. work focused on novel ranking problems in information networks. His research interests include topics in data mining and machine learning applied to complex networks analysis (e.g., social media networks, trust networks, semantic networks, bibliographic networks) and to remote sensing analysis. On these topics he has coauthored journal articles and conference papers, organized workshops, presented tutorials at international conferences and developed practical software tools.