DEEP TEMPORAL CLUSTERING: FULLY UNSUPERVISED LEARNING OF TIME-DOMAIN FEATURES

Anonymous authors
Paper under double-blind review

ABSTRACT

Unsupervised learning of timeseries data is a challenging problem in machine learning. Here, we propose a novel algorithm, Deep Temporal Clustering (DTC), a fully unsupervised method, to naturally integrate dimensionality reduction and temporal clustering into a single end to end learning framework. The algorithm starts with an initial cluster estimates using an autoencoder for dimensionality reduction and a novel temporal clustering layer for cluster assignment. Then it jointly optimizes the clustering objective and the dimensionality reduction objective. Based on requirement and application, the temporal clustering layer can be customized with any temporal similarity metric. Several similarity metrics are considered and compared. To gain insight into features that the network has learned for its clustering, we apply a visualization method that generates a heat map of regions of interest in the timeseries. The viability of the algorithm is demonstrated using timeseries data from diverse domains, ranging from earthquakes to sensor data from spacecraft. In each case, we show that our algorithm outperforms traditional methods. This performance is attributed to fully integrated temporal dimensionality reduction and clustering criterion.

1 INTRODUCTION

Deep learning has become the dominant approach to supervised classification problems where the network is trained on expert labeled static data (LeCun et al., 2015; Schmidhuber, 2015). However, in a large class of problems, either the expert labels are not reliable or they do not exist at all. To address such problems, a long list of clustering algorithms have been developed to organize data into separate categories. While much of the research has been devoted to techniques dealing with static data, there is a lack of machine learning algorithms for unsupervised timeseries data.

Accurate temporal clustering is essential in many science and engineering applications, such as event detection, financial trading, and medical applications (Aghabozorgi et al., 2015). However, standard clustering algorithms were designed for static data and do not account for temporal structure in the data. As such, they cannot be directly applied to timeseries. A number of factors make the clustering of the temporal data a particularly challenging task. Timeseries from different domains exhibit considerable variations in their properties, features, and temporal scales, and are generally high-dimensional. Further, timeseries data from real world applications often have data gaps as well as high fluctuating noise due to the data acquisition defects or the inherent nature of the data (Antunes & Oliveira, 2001).

To address these issues, we present a novel algorithm called deep temporal clustering (DTC) which transforms the timeseries into a latent space, that is compatible with any temporal similarity metric using a learnable deep network. This enables the neural net to learn simultaneously a compatible latent space and a clustering criterion. The transformation of timeseries into a low dimensional latent space in DTC is achieved through a deep autoencoder that is in turn fully integrated with a novel temporal clustering layer. The overview of our method is illustrated in Figure 1.

The network described in this work was designed based on several key observations. First, natural signals in time domain are continuous (i.e., nearby points in time are predictive of each-other) and can be decomposed into a relatively low number of dominant waveforms on a short time scale. Hence we use the convolutional layer (CNN) as a first component of our network to reduce dimensionality and to cast the temporal input into a more convenient spatio-temporal representation

without losing much relevant information. Second, natural signals in time domain may contain information over long time scale, potentially much longer than the time extent of the CNN filters. Hence we use a bidirectional LSTM to extract and represent those long-term features in the input. By performing joint end-to-end optimization of CNN and LSTM we achieve best clustering performance along relevant temporal or spatio-temporal dimensions for a broad range of time scales of relevant temporal features. Please note that here, for clarity, we only show results for single-channel temporal inputs. In a separate work we will present the results for multichannel inputs, replacing 1D CNN with 2D CNN.

To the best of our knowledge, this is the first work on the application of deep learning in temporal clustering. In the remainder of the paper, we demonstrate how DTC addresses the following questions:

1. What feature space to use?
2. How to integrate temporal clustering functionality into deep learning?
3. Can one visualize the key features in the timeseries that the network learned to use in its clustering result?
4. How to achieve an optimal result?

The main contribution of our study involves formulating an end-to-end deep learning algorithm, and implements objective formulation so as to achieve a temporal clustering. We carefully formulated the objective encompassing two crucial aspects essential for high clustering accuracy, i.e., an effective latent representation and a similarity metric which can be integrated into the learning structure. Joint optimization of these objectives together was able to produce a significant improvement in comparison with the baseline metrics. This is substantiated by extensive experimentation involving various real world datasets.

In autoencoder, we use a 1D CNN to extract temporal features, then use a Bidirectional LSTM to cast the temporal information along both directions into a reduced latent space.

We have conducted extensive numerical experiments using various temporal similarity metrics. Specifically, we studied 4 different metric combinations on publicly available UCR timeseries datasets as well as recent spacecraft data collected from NASA's Magnetospheric Multiscale (MMS) Mission. Our numerical experiments suggest that our proposed method improves the performance compared to baseline metrics on all datasets.

Finally, our proposed method presents some desirable features when applied to real clustering problems. Notably, it can visualize the activations which lead to cluster assignment. Most of the traditional clustering algorithms are not able to do this. This can be particularly useful for datasets where there is a need to localize events without labels.

2 Related Work

Much of the existing research to solve temporal clustering has focused on addressing one of the two core issues. First is an effective dimensionality reduction and second is choosing an effective similarity metric.

Several temporal clustering methods based on application dependent transformations have been previously proposed like adaptive piecewise constant approximation (Keogh et al., 2001) and nonnegative matrix factorization (Brockwell & Davis, 2013). Given the high-dimensional nature of timeseries, the transformations consist of dimensionality reduction approaches which can filter out the high fluctuating noise. However, these methods are inefficient because they perform dimensionality reduction independent of the clustering criterion. Moreover, these methods may lose long temporal correlation and even filter out some relevant feature. Furthermore, most of the existing strategies are based on hand crafted transformations which are meticulously designed for a specific application. This requires extensive knowledge of the data, which is not always possible. Recently, Xie et al. (2016) and Yang et al. (2016) achieved superior performance by jointly optimizing a stacked autoencoder for dimensionality reduction and a kmeans objective for clustering. However, both the stacked autoencoder and the clustering objective are designed for static data, i.e., they collapse any temporal component and will fail to cluster temporal data.

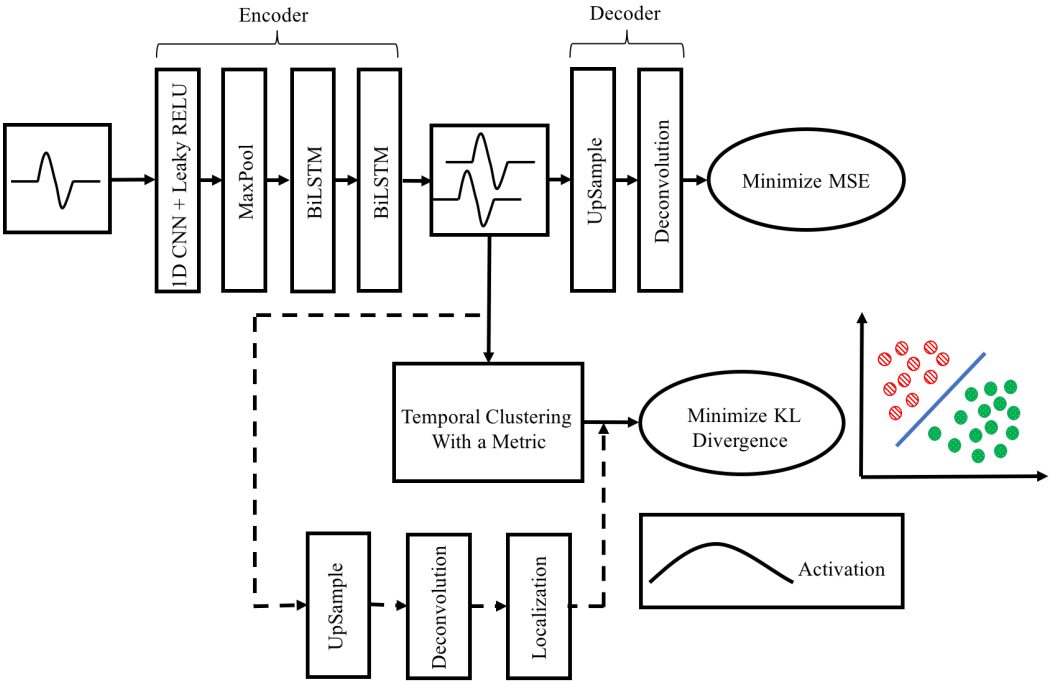


Figure 1: The overview of proposed DTC. The input signal is encoded into a latent space, which is then fed to a temporal clustering layer, thereby giving out the cluster assignments. The dashed lines indicate the heat map generating network.

Another approach has been based on creation of suitable similarity measure between two timeseries which take into account unique features like complexity (Batista et al., 2011), correlation (Galeano & Peña, 2000; Golay et al., 1998), and time warping (Berndt & Clifford, 1994). Using these metrics, traditional clustering algorithms such as k-means or hierarchical clustering were implemented. Recently, Montero et al. (2014) conducted a detailed study on various state-of-the-art similarity metrics. The choice of the appropriate similarity metric is a key step in these clustering approaches. However, similarity metric itself may not be sufficient for good clustering because of the complexity and high dimensional nature of temporal data.

The above-mentioned studies have shown that casting the original timeseries into a low dimensional feature space is well suited for timeseries clustering. However, these approaches lack a general method for the selection of the latent space that best captures the properties of a given timeseries data. Another key step in achieving optimal clustering results is ensuring the similarity metric is compatible with the feature space.

3 Proposed Method: Deep Temporal Clustering

Given a dataset of n temporal signals $X = x_1, x_2, .., x_n$, we group time series into k clusters based on a similarity metric.

3.1 Latent Transformation

Latent transformation is a key aspect of the temporal clustering. We achieve this by making use of a temporal autoencoder (TAE) as shown in Figure 1. The network architecture consists of a 1D convolution layer, which extracts key features, followed by max pooling which shortens the timeseries. This shortening of signal is crucial for further processing to avoid very long sequences which can lead to poor performance. We use leaky rectified linear units (ReLUs) as activation function. Convolutional features are then fed to Bi-LSTM's to obtain the latent representation. We use Bidirectional LSTM so that they can perceive temporal changes along both directions and

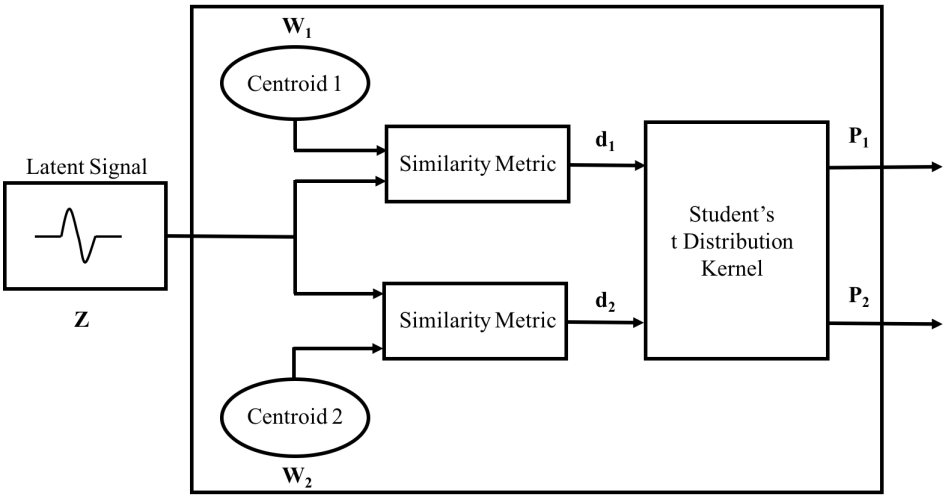


Figure 2: Temporal clustering layer for a 2 cluster problem.

can collapse the input sequences in all dimensions except temporal and can cast the input into a much smaller latent space. Finally, the latent representation is decoded using upsampling and a deconvolutional layer. Architectures having convolutional features followed by LSTM produced consistently good results in supervised speech applications (Hannun et al., 2014) and so we adapt this into our fully unsupervised framework. Note that for a specific application, one may achieve higher accuracy with some modification of our architecture, but the motivation of this study is focus on a general framework which can encompass a broad range of timeseries data. To avoid overfitting, we incorporate dropout (Witten et al., 2016) during the optimization. Training is performed by minimizing the L2 loss.

3.2 Temporal Clustering Layer

The temporal clustering layer consists of k centroids having weights of $w_j, j \in 1..k$. To initialize these cluster centroids, we use the latent signals z_i obtained by feeding input $x_i \in X$ through the initialized TAE. z_i are then used to perform hierarchical clustering in the feature space Z through a similarity metric (discussed in the next section 3.2.2). This yields the initial centroids estimates $w_j, j \in 1..k$, where k is number of clusters.

3.2.1 Clustering Criterion

After we obtain an initial estimate of the centroids w_j, we train the temporal clustering layer using an unsupervised algorithm that alternates between two steps.

1. First, we compute the probability of assignment of input x_i belonging to the cluster j. The closer the input x_i is to the centroid w_j, the higher the probability belonging to cluster j
2. Second, we update the centroids by using a loss function, which maximizes the high confidence assignments using an external distribution.

These two steps are iterated until we reach a stop condition, which in our case is when the change in the centroids is less than 1% or some maximum iterations.

3.2.2 Clustering Assignment

When an input z_i is fed to the temporal clustering layer, we compute distances d_{ij} from each of the centroids w_j using a similarity metric. Note that the similarity computation is inside the neural network and fully integrated in the learning process. Then we normalize the distances d_{ij} into probability assignments using a Student's t distribution kernel (Maaten & Hinton, 2008). The probability

assignment of latent signal belonging to k^{th} cluster is as follows:

$$q_{ij} = \frac{\left(1 + \frac{siml(z_i, w_j)}{\alpha}\right)^{-\frac{\alpha+1}{2}}}{\sum_{j=1}^{k}\left(1 + \frac{siml(z_i, w_j)}{\alpha}\right)^{-\frac{\alpha+1}{2}}} \tag{1}$$

Here q_{ij} is the probability of input i belonging to cluster j, z_i corresponds to the signal in the latent space Z, obtained from temporal autoencoder after encoding the input signal $x_i \in X$. The parameter α is the number of degrees of freedom of the Students t distribution. In an unsupervised setting, since there is no actual learning, we can set $\alpha = 1$ as suggested by (Maaten & Hinton, 2008). Lastly $siml()$ is the temporal similarity metric which is used to compute the distance between the encoded signal z_i and centroid w_j. We illustrate a 2 cluster example in Figure 2, where the latent input z is used to compute distances d_1 and d_2 from centroids w_1 and w_2 using a similarity metric, Later converted into probabilities p_1 and p_2 using a Students t distribution kernel.

In this study, we experiment with various similarity metrics as follows:

1. **Invariant Similarity(CID)** proposed by Batista et al. (2011) will compute the similarity based on the euclidean distance which is corrected by complexity estimation of the two series. This distance is calculated as follows:

$$(x, y) = ED(x, y)CF(x, y) \tag{2}$$

where $CF(x, y)$ is a complexity factor defined as follows: $\frac{max(CE(x),CE(y))}{min(CE(x),CE(y))}$ and $CE(x)$ and $CE(y)$ are the complexity estimates of a timeseries x and y, respectively. The core idea of CID is that with increase in the complexity differences between the series, the distance increases. Further, if both input sequences have the same complexity then the distance simply is the euclidean distance. The complexity of each sequence is defined as:

$$CE(x) = \sqrt{\sum_{t=1}^{N-1}(x_{t+1} - x_t)^2} \tag{3}$$

where N is length of the sequence

2. **Correlation based Similarity(COR)** as used by Golay et al. (1998), computes similarities using the estimated pearsons correlation ρ between the latent representation z_i and the centroids w_j . In this study we compute the COR as follows:

$$COR = \sqrt{2(1 - \rho)} \tag{4}$$

where ρ is the pearson's correlation, given by $\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$ and cov is the covariance.

3. **Auto Correlation based Similarity(ACF)** as used in Galeano & Peña (2000), computes the similarity between the latent representation z_i and the centroids w_j using autocorrelation (ACF) coefficients and then performs weighted euclidean distance between the autocorrelation coefficients.

4. **Euclidean(EUCL)** is the euclidean distance between the input signal and each of the centroids.

3.2.3 Clustering Loss Computation

Finally, to train the temporal clustering layer iteratively we need to compute a loss. After we obtain the probability assignments, we formulate the objective as minimizing the KL divergence loss between q_{ij} and a high confidence target distribution so as to maximize high confidence assignments. This is realized using a target distribution p_{ij}

$$p_{ij} = \frac{q_{ij}^2 / f_j}{\sum_{j=1}^{k} q_{ij}^2 / f_j} \tag{5}$$

where $f_j = \sum_{i=1}^{n} q_{ij}$. Further empirical properties of this distribution were discussed in detail in Hinton et al. (2006) and Xie et al. (2016). Now using this high confidence distribution, we compute the KL divergence loss using the following equation:

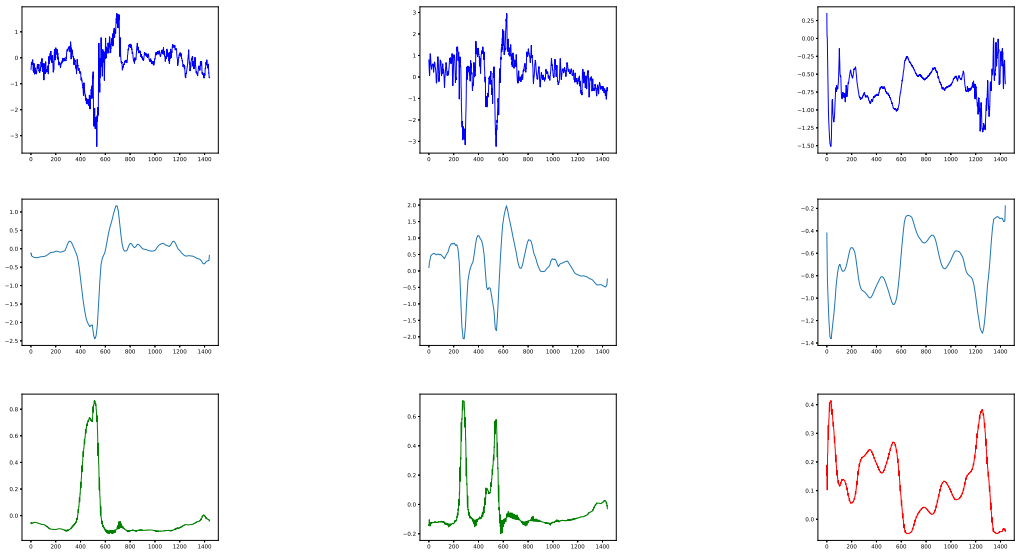
$$L = \sum_{i=1}^{n} \sum_{j=1}^{k} p_{ij} log \frac{p_{ij}}{q_{ij}} \tag{6}$$

where n and k are number of samples in dataset and number of clusters respectively. Since we are maximizing the high confidence assignments, this distribution p works robustly when the initial estimates are reliable. Therefore, the choice of metric used for clustering becomes so important.

3.3 DTC Optimization

We perform batch wise joint optimization of clustering and autoencoder by minimizing KL divergence loss and mean squared error loss, respectively. Similar approaches have been used in Xie et al. (2016) and Yang et al. (2016). This helps prevent any problematical solutions from drifting too far away from the original input signal. Because there is also an objective to reconstruct the original signal, the latent representation will converge at suitable representation so as to minimize both the clustering loss and the dimensionality reduction loss.

3.4 Visualizing Heat Map



One Event | Two Events | Non-Event

Figure 3: The top panel in each column represents the input timeseries. Results for three input timeseries are shown. The middle panels correspond to the decoded output and the bottom panel represents the heat map. The green and red in the heat map represent prediction of event and non-event, respectively. The first two columns contain a timeseries with one and two events, respectively. The third column is a non-event, correctly identified by our algorithm as it is marked in red.

In this work, we had a kin interest in being able to localize the main feature(s) that triggers the final classification. For example, in the NASA data set 4.1, we had the knowledge that the bi-polar feature is the main trigger for a time series to be classified as an event 3. Adapting a similar heat map structure that has been used to localize tumors using image level labels (Hwang & Kim, 2016), we were able to localize bi-polar features successfully. The most important difference here is our network is fully unsupervised and for the training we make use of the loss generated by temporal cluster layer as discussed in 3.2.3. In the visualization network shown by the dashed lines in Fig.1, we feed the latent representation z_i to the upsampler, followed by a 1D deconvolutional layer, a dense layer and

the results finally added to the temporal clustering output. During training, this layer is optimized along with the temporal clustering using the same loss as Eq. 6. It is worth reiterating that the loss used in (Hwang & Kim, 2016) is computed using a weighted addition of classification and localization loss. Whereas in our work, we add the visualization and temporal clustering outputs to compute the loss L using Eq. 6, thereby integrating visualization to our unsupervised framework. Optimizing this structure by incorporating various merge techniques and hyper-parameters can improve the localization performance and that can be addressed in a separate work.

4 EXPERIMENTS

DTC Experiments are performed using tensorflow 1.3 and keras 2.0 software on Nvidia GTX 1080Ti graphics processor. The Baseline algorithms are performed in R using publicly available TSclust Package (Montero et al., 2014).

4.1 DATASETS

The performance of the proposed algorithm is evaluated on a wide range of real world datasets.

- One dataset is from measurements by spacecraft magnetometer from the recent NASA Magnetospheric Multiscale (MMS) Mission. The magnetospheric plasma environment consists of many transient structures and waves. Here we are interested in automated detection of spacecraft crossings of the so-called flux transfer events (FTEs) in an otherwise turbulent medium. FTEs (Russell & Elphic, 1979) are characterized by bipolar signature in the B_N component of the magnetic field. The dataset contains a total of 104 time series, with each sequence having 1440 time steps.
- We also make use of a few publicly available UCR Time series Classification Archive datasets (Chen et al., 2015). Table 1 includes the properties of these datasets where we describe three attributes for each dataset. This includes the number of samples N, the time steps in each sequence L, and the class distribution ratio r. Since we are using these datasets as unlabeled data, we combine the training and test datasets for all UCR datasets for all experiments in this study.

4.2 BASELINE METHODS

We compare the results of the method, hierarchical clustering with complete linkage, against our algorithm (DTC) for the following metrics:

1. *Complexity Invariant Distance(CID)*
2. *Correlation based Similarity(COR)*
3. *Auto Correlation based Similarity(ACF)*
4. *Euclidean Based Similarity(EUCL)*

4.3 EVALUATION METRICS

Since we have labels for the timeseries, we can compare the results of our clustering with the expert labels. This provides a gauge of accuracy of our unsupervised algorithm as a classifier. To this end, we use the Receiver Operating Characteristics (ROC) (Fawcett, 2006) and area under the curve (AUC) to assess the efficacy of our approach. We emphasize that our entire training pipeline is unsupervised and we only use the labels to measure the performance of our algorithm on real world datasets. We used bootstrap sampling (Singh & Xie, 2008) and averaged the ROC curves over 5 trials.

4.4 PARAMETER INITIALIZATION

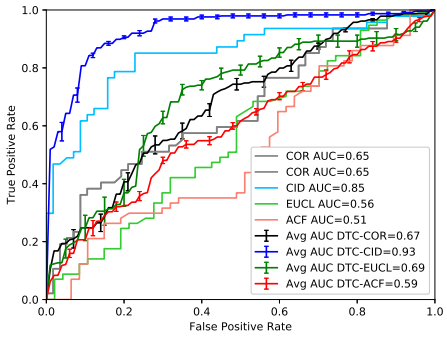
Autoencoder network is initialized with 10 epochs and temporal clustering layer centroids are initialized using hierarchical clustering with complete linkage and using the respective metrics obtained from the TSclust package. We use the Adam optimizer for our network training.

4.5 QUALITATIVE ANALYSIS

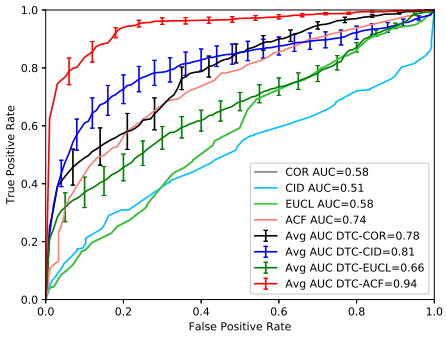
In Fig. 3, each column shows the results obtained in various stages after feeding the input signal through the network. The first row represents the input signal, the second row represents the autoencoder output. As illustrated in Fig. 3 the autoencoder gets rid of the chronic noise present in the input signal and smooths the signal. The last row represents the activation map generated by the algorithm (green and red represent an event prediction and non-event prediction, respectively). As shown in the Fig. 3, the algorithm was able to capture the bipolar signal in column 1 and column 2. Further, column 2 represents a multi-event input, in which our algorithm was able to capture both events. The column 3 is a non-event and is correctly identified as such by our algorithm (no green output).

4.6 QUANTITATIVE ANALYSIS

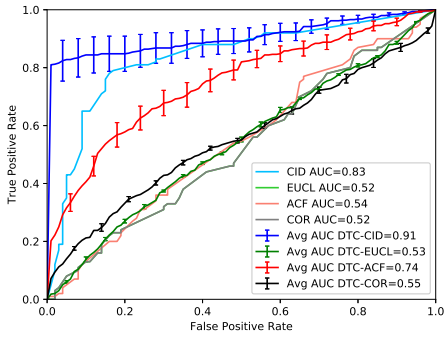
We have chosen 4 different metrics over a variety of 13 datasets. The Fig. 4 illustrates the comparison of clustering performance on 4 different datasets over 4 different metrics. We can see that our algorithm was able to improve the baseline performance in all of the datasets over all metrics. More detailed comparison is shown in table 1. Our algorithm shows to be robust and obtained improved results across timeseries with different dataset sizes N, different lengths L, as well as a range of data imbalances denoted by r.



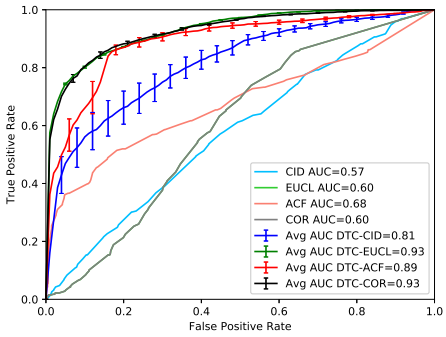
(a) NASA MMS Dataset



(b) UCR SonyAIBORobotSurface Dataset



(c) UCR ShapeletSim Dataset



(d) UCR MoteStrain Dataset

Figure 4: Performance of our algorithm compared to the baseline metrics. The ROC curves of our proposed method are averaged over 5 trials. The mean ROC and standard error bars are shown. Baseline metrics are averaged over 10 runs using bootstrap sampling.

Table 1: The performance of our algorithm compared to the baseline metrics. Each entry represents the AUC averaged over 5 trials. Baseline metrics AUC's are averaged over 10 runs using bootstrap sampling. The first column indicates dataset size N, time series length L and class distribution $r = \frac{\text{no pos class}}{\text{no neg class}}$

Dataset(N,L,r)	ACF	DTC ACF	CID	DTC CID	COR	DTC COR	EUCL	DTC EUCL
BeetleFly (40,512,1.00)	0.55	0.69	0.8	0.892	0.55	0.584	0.55	0.606
BirdChicken (40,512,1.00)	0.7	0.792	0.6	0.732	0.55	0.712	0.55	0.772
Computers (500,720,1.00)	0.58	0.64	0.51	0.68	0.55	0.555	0.55	0.58
Earthquakes (461,512,0.25)	0.508	0.569	0.508	0.588	0.546	0.549	0.546	0.540
MoteStrain (1272,84,0.86)	0.68	0.89	0.57	0.81	0.60	0.93	0.60	0.93
Phalanges OutlinesCorrect (2658,80,1.77)	0.586	0.522	0.501	0.529	0.501	0.525	0.501	0.556
Proximal PhalanxOutlineCorrect (891,80,2.12)	0.52	0.678	0.5	0.66	0.52	0.62	0.52	0.63
ShapeletSim (200,500,1.00)	0.54	0.74	0.83	0.91	0.52	0.55	0.52	0.53
SonyAIBO RobotSurfaceII (980,65,1.61)	0.56	0.72	0.827	0.85	0.57	0.82	0.57	0.83
SonyAIBO RobotSurface (621,70,0.78)	0.74	0.94	0.51	0.81	0.58	0.78	0.58	0.66
ItalyPowerDemand (1096,24,1)	0.59	0.63	0.60	0.66	0.54	0.57	0.54	0.61
WormsTwoClass (258,900,1.37)	0.53	0.62	0.59	0.61	0.55	0.56	0.55	0.51

5 DISCUSSION

Our work can be extended in several directions. First, not every temporal metric can be efficiently computed in the deep learning pipeline. For instance, metrics like Dynamic Time Warping whose computation time is $O(N^2)$, will be expensive to integrate in our temporal clustering layer. Effective implementation strategies can be explored.

The proposed algorithm is sensitive to initialization. If the baseline metric performs poorly, the improvement using DTC may not be high. Since we maximize the high confidence assignments, we have to start with at least a few high confidence assignments.

Finally, the aim of this study was to propose a general network for fully unsupervised learning of time-domain features and the associated clustering. It may be possible to modify the network or the optimization of the temporal autoencoder component and the visualization component, depending on the size and properties of the dataset.

6 CONCLUSION

In this work we addressed the question of unsupervised learning of patterns in temporal sequences, unsupervised event detection and clustering. Post-hoc labeling of the clusters, and comparison to ground-truth labels not used in training, reveals high degree of agreement between our unsupervised

clustering results and human-labeled categories, on several types of datasets. This indicates graceful dimensionality reduction from inputs with extensive and complex temporal structure to one or few-dimensional space spanned by the cluster centroids. As most natural stimuli are time-continuous and unlabeled, the approach promises to be of great utility in real-world applications. Generalization to multichannel spatio-temporal input is straightforward and has been carried out as well; it will be described in a separate paper.

REFERENCES

Saeed Aghabozorgi, Ali Seyed Shirkhorshidi, and Teh Ying Wah. Time-series clustering–a decade review. *Information Systems*, 53:16–38, 2015.

Cláudia M Antunes and Arlindo L Oliveira. Temporal data mining: An overview. In *KDD workshop on temporal data mining*, volume 1, pp. 13, 2001.

Gustavo EAPA Batista, Xiaoyue Wang, and Eamonn J Keogh. A complexity-invariant distance measure for time series. In *Proceedings of the 2011 SIAM International Conference on Data Mining*, pp. 699–710. SIAM, 2011.

Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pp. 359–370. Seattle, WA, 1994.

Peter J Brockwell and Richard A Davis. *Time series: theory and methods*. Springer Science & Business Media, 2013.

Yanping Chen, Eamonn Keogh, Bing Hu, Nurjahan Begum, Anthony Bagnall, Abdullah Mueen, and Gustavo Batista. The ucr time series classification archive, July 2015. `www.cs.ucr.edu/~eamonn/time_series_data/`.

Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

Pedro Galeano and Daniel Pella Peña. Multivariate analysis in vector time series. *Resenhas do Instituto de Matemática e Estatística da Universidade de São Paulo*, 4(4):383–403, 2000.

Xavier Golay, Spyros Kollias, Gautier Stoll, Dieter Meier, Anton Valavanis, and Peter Boesiger. A new correlation-based fuzzy logic clustering algorithm for fmri. *Magnetic Resonance in Medicine*, 40(2):249–260, 1998.

Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

Sangheum Hwang and Hyo-Eun Kim. Self-transfer learning for fully weakly supervised object localization. *arXiv preprint arXiv:1602.01625*, 2016.

Eamonn Keogh, Kaushik Chakrabarti, Michael Pazzani, and Sharad Mehrotra. Locally adaptive dimensionality reduction for indexing large time series databases. *ACM Sigmod Record*, 30(2): 151–162, 2001.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008.

Pablo Montero, José A Vilar, et al. Tsclust: An r package for time series clustering. *Journal of*, 2014.

Christopher T Russell and RC Elphic. Isee observations of flux transfer events at the dayside magnetopause. *Geophysical Research Letters*, 6(1):33–36, 1979.

Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

Kesar Singh and Minge Xie. Bootstrap: a statistical method. 2008.

Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International Conference on Machine Learning*, pp. 478–487, 2016.

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. *arXiv preprint arXiv:1610.04794*, 2016.