Unsupervised Representation Learning for Time Series with Temporal Neighborhood Coding

Anonymous authors
Paper under double-blind review

Abstract

Time series are often complex and rich in information, but sparsely labeled and therefore challenging to model. In this paper, we propose a self-supervised framework for learning robust and generalizable representations for time series. Our approach, called Temporal Neighborhood Coding (TNC), takes advantage of the local smoothness of a signal's generative process to define neighborhoods in time with stationary properties. Using a debiased contrastive objective, our framework learns time series representations by ensuring that in the encoding space, the distribution of signals from within a neighborhood is distinguishable from the distribution of non-neighboring signals. Our motivation stems from the medical field, where the ability to model the dynamic nature of time series data is especially valuable for identifying, tracking, and predicting the underlying patients' latent states in settings where labeling data is practically impossible. We compare our method to recently developed unsupervised representation learning approaches and demonstrate superior performance on clustering and classification tasks for multiple datasets.

1 Introduction

Real world time-series data is high dimensional, complex, and has unique properties that bring about many challenges for data modeling (Yang & Wu, 2006). In addition, these signals are often sparsely labeled which makes it even more challenging for supervised learning tasks. Unsupervised representation learning can extract informative low-dimensional representations from raw time series by leveraging the inherent structure of the data, without the need for explicit supervision. These representations are more generalizable and robust to noise, as they are less specialized for solving a single supervised task. Unsupervised representation learning is well studied in domains such as vision (Donahue & Simonyan, 2019; Denton et al., 2017; Radford et al., 2015) and natural language processing (Radford et al., 2017; Young et al., 2018; Mikolov et al., 2013), but has been underexplored in the literature for time series settings. Frameworks designed for time series need to be efficient and scalable, given that signals encountered in practice can be long, high dimensional and even high frequency. Moreover, it should account for and be able to model dynamic changes that occur within samples, i.e. non-stationarity of signals.

The ability to model the dynamic nature of time series data is especially valuable in medicine. Health care data is often organized as a time series, with multiple data types, collected from a variety of sources at different sampling frequencies, and riddled with artefacts and missing values. Patients, throughout their stay at the hospital or within the disease progression period, transition gradually between distinct clinical states. A particular challenge in medical time-series data is the lack of well-defined or available labels that are needed for identifying the underlying clinical state or for training models aimed at extracting low-dimensional representations of these states. For instance, in the context of critical-care, a patient's stay in the critical care unit (CCU) is captured continuously via streaming physiological signals by the bedside monitor. Obtaining labels for the patient's state for extended periods of these signals is practically impossible as the underlying physiological state can be unknown even to the clinicians. This further motivates the use of unsupervised representation learning in these contexts. Learning low dimensional, rich, and robust unsupervised representations can be crucial in facilitating the tracking of disease progression, predicting the future trajectories of the patients, and tailoring treatments to these underlying states.

In this paper, we propose a self-supervised framework for learning representations for complex multivariate non-stationary time series. This approach, called Temporal Neighborhood Coding (TNC), is designed for temporal settings where the latent distribution of the signals changes over time, and it aims to capture the progression of the underlying temporal dynamics. TNC is efficient, easily scalable to high dimensions, and can be used in different time series settings. We assess the quality of the learned representations on multiple datasets and show that the representations are general, transferable, and can be used for a number of downstream tasks such as classification and clustering. We further demonstrate that our method outperforms existing approaches for unsupervised representation learning, and it even performs closely to supervised techniques in classification tasks. Our contributions are three-fold:

1. We present a novel neighborhood-based unsupervised learning framework for *non-stationary* multivariate time series data.

2. We introduce the concept of a temporal neighborhood as the distribution of similar windows in time. The neighborhood boundaries are determined automatically using the properties of the signal and statistical testing.

3. We incorporate concepts from Positive Unlabeled Learning, specifically, sample weight adjustment, to account for potential bias introduced in sampling negative examples for contrastive loss.

2 METHOD

We introduce a framework for learning low dimensional representations that encode the underlying state of a multivariate, non-stationary time series. Our self-supervised approach, TNC, takes advantage of the local smoothness of the generative process of signals to learn generalizable representations for windows of time series. This is done by ensuring that in the representation space, the distribution of signals proximal in time is distinguishable from the distribution of signals far away, i.e. proximity in time is identifiable in the encoding space. We represent our multivariate time series signals as $X \in R^{D\times T}$, where D is the number of features and T is the number of measurements in time. $X_{[t-\frac{\delta}{2},t+\frac{\delta}{2}]}$ represents a window of time series of length δ, centered around time t, that includes measurements of all features taken in the interval $[t-\frac{\delta}{2},t+\frac{\delta}{2}]$. Throughout the paper, we refer to this window as W_t for notational simplicity.

We define the temporal neighborhood (N_t) of a window W_t as the set of all windows with centroids t^*, sampled from a normal distribution $t^* \sim \mathcal{N}(t, \eta\cdot\delta)$. Where $\mathcal{N}$ is centered at t, δ is the size of the window, and η is the parameter that defines the range of the neighborhood. The neighborhood distribution is characterized as a Gaussian to model the gradual transition in temporal data, and intuitively, approximates the distribution of samples that are similar to W_t. The η parameter determines the neighborhood range, and depends on the signal characteristics and how gradual the statistical properties of the time series change over time. This can be set by domain experts based on prior knowledge of the signal behavior, or for a more robust estimation, it can be determined by analysing the stationarity properties of the signal, for every W_t. Since the neighborhood represents similar samples, the range should identify the approximate time span within which the signal remains stationary and the generative process doesn't change. For this purpose, we use the Augmented Dickey Fuller statistical test to determine this region. More details on this test and how it can be used to estimate η is described in section 2.

Now, assuming windows within a neighborhood possess similar properties, signals outside of this neighborhood, denoted as $\bar{N}_t$, are considered non-neighboring windows. Samples from $\bar{N}_t$ are likely to be different from W_t, and can be considered as negative samples in a context of a contrastive learning framework. However, this can suffer from the problem of *sampling bias*, common in most contrastive approaches (Chuang et al., 2020; Arora et al., 2019). Drawing negative examples from the data distribution may result in negative samples that are actually similar to the reference, which is W_t in our context. This can significantly impact the performance of the learning framework, but limited work has been done on addressing this issue (Chuang et al., 2020). To alleviate this bias in the TNC framework, we consider samples from $\bar{N}_t$ as unlabeled samples, as opposed to negative, and use ideas from Positive-Unlabeled (PU) learning to accurately measure the loss function. In reality, even though samples within a neighborhood are all similar, we cannot make the assumption that samples outside this region are necessarily different. For instance, in the presence of long term seasonalities, signals can exhibit similar properties at distant times. In a healthcare context, this can be like a stable patient that undergoes a critical condition, but returns back to a stable state afterwards. In PU learning, a classifier is learned using labeled data (drawn from the positive class) and unlabeled data that is a mixture of positive and negative samples with a positive class prior π (Du Plessis et al., 2014; Kiryo et al., 2017; Du Plessis & Sugiyama, 2014). Existing PU methods fall under two categories based on how they handle the unlabeled data: 1) methods that identify negative samples from the unlabeled cohort (Li & Liu, 2003); 2) methods that treat the unlabeled data as negative samples with smaller weights (Lee & Liu, 2003; Elkan & Noto, 2008). In the second category, unlabeled samples should be properly weighted in the loss term in order to train an unbiased classifier. Elkan & Noto (2008) introduces a simple and efficient way of approximating the expectation of a loss function by assigning individual weights w to training examples from the unlabeled cohort. This means each sample from the neighborhood is treated as a positive example with unit weight, while each sample from $\bar{N}$ is treated as a combination of a positive example with weight w and a negative example with complementary weight $1 - w$. In the original paper (Elkan & Noto, 2008), the weight is defined as the probability of having a positive sample in the unlabeled set, i.e. $w = p(y = 1|x \in U)$. In the TNC framework, this weight represents the probability of having samples similar to W_t in $\bar{N}$. By incorporating weight adjustment into the TNC loss (Equation 1), we account for possible positive samples that occur in the non-neighboring distribution. w can be approximated using heuristics or prior knowledge of the underlying state distribution, or tuned as a hyper parameter. Appendix A.5 explains how the weight parameter is selected for our different experiments, and also demonstrates how adding this weight adjustment improves the performance for downstream tasks.

After defining the neighborhood distribution, we can train an objective function that encourages distinction between the representation of samples from the same neighborhood from the outside samples. Assume $p(W_l \in N_t)$ encodes the probability of W_l being in the neighborhood of W_t. For an ideal encoder that preserves the neighborhood properties in the encoding space, $p(Z_l \in N_t)$ should be close to $p(W_l \in N_t)$, where Z_l is the representation of W_l. TNC has two main components:

1. An Encoder $Enc(W_t)$ that maps a window $W_t \in \mathbb{R}^{D\times\delta}$ to a representation $Z_t \in \mathbb{R}^M$, in a lower dimensional space ($M \ll D \times \delta$), where $D \times \delta$ is the total measurements in W_t.
2. A Discriminator $\mathcal{D}(Z_t, Z_l)$ that approximates the probability $p(Z_l \in N_t)$. More specifically, it receives two samples from the encoding space and predicts the probability of those samples being in the same temporal neighborhood.

Since TNC is a general framework, agnostic to the nature of the time series, the encoder can be modeled with any parametric model well-suited to the signal properties (Oord et al., 2016; Bai et al., 2018; Fawaz et al., 2019). The Discriminator $\mathcal{D}(Z_t, Z_l)$ is a simple multi-headed binary classifier that outputs 1 if Z_l and Z_t are representation of neighbors in time, and 0 otherwise. In the experiment section we describe the architectural details of the models used for our experiments in more depth.

Figure 1 provides a summary overview of the TNC framework. We formalize the objective function of our unsupervised learning framework in Equation 1. In essence, we would like the probability likelihood estimation of the Discriminator to be accurate, i.e. close to 1 for representation of neighboring samples, and close to 0 for windows far apart. Samples from the non-neighboring region ($\bar{N}$) are weight-adjusted using the w parameters, to account for positive samples in this distribution.

$$\mathcal{L} = -\mathbb{E}_{W_t\sim X}[\mathbb{E}_{W_l\sim N_t}[\log \underbrace{\mathcal{D}(Z_t, Z_l)}_{\mathcal{D}(Enc(W_t), Enc(W_l))}] + \mathbb{E}_{W_l\sim \bar{N}_t}[(1-w_t)\times(1-\log \underbrace{\mathcal{D}(Z_t, Z_l)}_{\mathcal{D}(Enc(W_t), Enc(W_l))}) - w_t \times \log \mathcal{D}(Z_t, Z_l)]] \tag{1}$$

We train the encoder and the discriminator hand in hand by optimizing for this objective. Note that the Discrimintor is only part of training, and will not be used during inference. Similar to the encoder, it can be approximated using any parametric model. However, the more complex the discriminator, the harder it becomes to interpret the latent space, since is allows similarities to be mapped on nonlinear complex relationships.

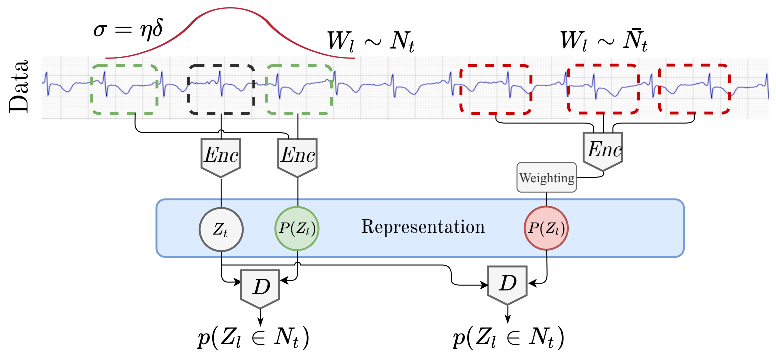


Figure 1: Overview of TNC framework components. The encoder learns the distribution of windows from N_t and $\bar{N}_t$, in the representation space. Then samples from these distributions are fed into the discriminator alongside Z_t, to predict $P(Z_l \in N_t)$.

Defining the neighborhood parameters using the ADF test: As mentioned earlier, the neighborhood range can be specified using the characteristics of the data. In non-stationary time series, the generative process of the signals changes over time, but we would like to define the temporal neighborhood such that the signal is stationary within it. A signal may remain in an underlying state for an unknown amount of time, therefore the neighborhood range must be adjusted to signal behavior at every point. To that end, we use the Augmented Dickey Fuller (ADF) statistical test to adjust the neighborhood range η. The ADF test belongs to a category of tests called "Unit Root Test", and is a method for testing the stationarity of a time series. For every W_t, we gradually increase the neighborhood size and measure the p-value from the test. Once p-value is above a threshold (in our setting 0.01), it means that it fails to reject the null hypothesis and suggest that within this window, the signal is no longer stationary. This way we find the widest neighborhood within which the signal remains relatively stationary.

3 EXPERIMENTS

We evaluate the usability of our framework on multiple time series datasets with dynamic latent states that change over time. We compare classification performance and clusterability against two state of the art approaches for unsupervised representation learning for time series: 1. Contrastive Predictive Coding (CPC) (Oord et al., 2018) that uses principles of predictive coding to train the encoder on a probabilistic contrastive loss. 2. Triplet-Loss (T-Loss), introduced in (Franceschi et al., 2019), which employs time-based negative sampling and a triplet loss to learn representations for time series windows. The triplet loss objective ensures similar time series have similar representations by minimizing the pairwise distance between positive samples (subseries) while maximizing it for negative ones. (See Appendix A.2 for more details on each baseline and its implementation.)

For a fair comparison and to ensure that the difference in performance is not due to the differences in the architecture of the models, the same encoder network is used across all compared baselines. We assess the generalizability of the representations by: 1) evaluating clusterability in the encoding space and 2) using the representations for a downstream classification task. In addition to the above mentioned baselines, we also compare clusterability performance with unsupervised K-means, and classification with a K-Nearest Neighbor classifier, using Dynamic Time Warping (DTW) to measure time series distance. All models are implemented using Pytorch 1.3.1 and trained on a machine with Quadro 400 GPU. Below we describe each dataset in more details:

3.1 SIMULATED DATA

The simulated dataset is designed to replicate very long, high-frequency time series for which underlying dynamics changes over time. Our generated time series consists of 2000 measurements for 3 features, generated from 4 different underlying states. We use a hidden Markov Model to generate the random latent states over time, and in each state, the time series are generated from a different generative process, including GPs with different kernel functions and Nonlinear Autoregressive Moving Average models with different sets of parameters (α and β). In addition, for it to further resemble realistic (e.g. clinical) time series, two features are always correlated. More details about this dataset is provided in the Appendix A.1. For this experimental setup, we use a two-directional, single-layer recurrent neural network encoder that encodes multi-dimensional signal windows of $\delta = 50$ into 10 dimensional representation vectors.

3.2 Clinical waveform data

For a real-world clinical experiment, we use the MIT-BIH Atrial Fibrillation dataset Moody (1983). This dataset includes 25 long-term ECG recordings (10 hours in duration) of human subjects with atrial fibrillation. It consists of two ECG signals each sampled at 250 Hz with rhythm annotations of types: 1) Atrial fibrillation, 2) Atrial flutter, 3) AV junctional rhythm, and 4) all other rhythms. Our goal in this experiment is to identify the underlying type of arrhythmia over time, for each sample without any information about the annotations. This dataset is an interesting experiment due to the following special properties:

- The underlying heart rhythm changes over time in each sample. This is an opportunity to evaluate how different representation learning frameworks can handle alternating classes;
- The dataset is highly imbalanced, with atrial flutter and AV junctional rhythm being present in fewer than 0.1% of the measurements. Data imbalance poses many challenges for downstream classification, further motivating the use of unsupervised representation learning;
- The dataset has samples from a small number of individuals, but over an extended period of time (around 5 million data points). This realistic scenario, common in healthcare data, shows that our framework can even be used in settings with a limited number of samples.

The encoder architecture Enc used in this experiment is a 2-channel, 1-dimensional strided convolutional neural network that runs directly on the ECG waveforms. We use six convolutional layers with a total down-sampling factor of 16. The window size is 2500 samples, meaning that each convolutional filter covers at least half a second of ECG recording, and the representations are summarized in a 64-dimensional vector.

3.3 Human Activity Recognition (HAR) data

Human Activity Recognition (HAR) is the problem of predicting the type of an activity using temporal data from accelerometer and gyroscope measurements. We use the HAR dataset from the UCI Machine Learning Repository [1] that includes data collected from 30 individuals, using a smart watch. Each person performs six activities: 1) walking, 2) walking upstairs, 3) walking downstairs, 4) sitting, 5) standing, and 6) laying. The time series measurements are pre-processed to extract 561 features. For our purpose, we concatenate activity samples from every individual over time using the subject identifier, in order to build the full time series for each subject, that includes continuous activity change. Similar to the simulated data setting, we use a single-layer RNN encoder. The selected window size is 4, which represents about 15 seconds of recording, and the representations are encoded in a 10-dimensional vector space.

4 Results

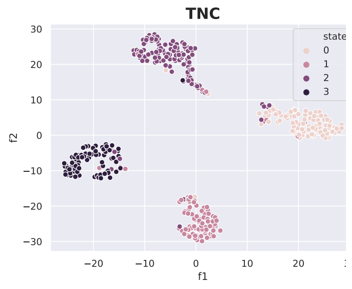
In this section we present performance results for clusterability and classification on all datasets. Clusterability indicates how well each method recovers appropriate states, and classification assesses how informative our representations are for downstream tasks.

4.1 Evaluation: Clusterability

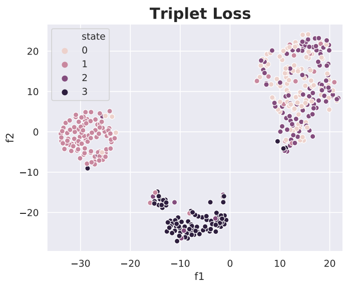
Many real-world time series data have underlying multi-category structure, naturally leading to representations with clustering properties. Encoding such general priors is a property of a good representation (Bengio et al., 2013). In this section, we assess the distribution of the representations

[1] https://archive.ics.uci.edu/ml/datasets/human+activity+recognition+using+smartphones

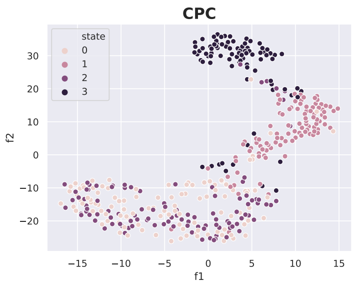
in the encoding space. If information of the latent state is properly learned and encoded by the framework, the representations of signals from the same underlying state should cluster together. Figures 2a, 2b, and 2c show an example of this distribution for simulated data across compared approaches. Each plot is a 2 dimensional t-SNE visualization of representations where each point in the scatter plot is an encoding $Z \in R^{10}$ that represents a window of $\delta = 50$ of a simulated time series. We can see that without any information about the hidden states, representations learned using TNC cluster windows from the same hidden state better than the alternatives. Results show that CPC and Triplet Loss have difficulty separating time series generated from non–linear auto-regressive moving average (NARMA) models with variable regression parameters.



(a) TNC representations



(b) T-loss representations



(c) CPC representations

Figure 2: T-SNE visualization of signal representations for the simulated dataset across all baselines. Each point in the plot is a 10 dimensional representation of a window of time series, with the color indicating the latent state. See Appendix A.6 for more plots from different datasets.

To compare the consistency of the representation clusters formed for each baseline we use two very common cluster validity indices, namely, Silhouette score and Davies-Bouldin index. We use K-means clustering in the representation space to measure these clusterability scores. The Silhouette score measures the similarity of each sample to its own cluster, compared to other clusters. The values can range from -1 to $+1$, and a greater score implies better cohesion. The Davies-Bouldin Index measures intra-cluster similarity and inter-cluster differences. This is a positive index score, where smaller values indicate low within-cluster scatter and large separation between clusters, i.e. a lower score is better (more details on the cluster validity scores can be found in Appendix A.4).

	Simulation		ECG Waveform		HAR	
Method	Silhouette ↑	DBI ↓	Silhouette ↑	DBI ↓	Silhouette ↑	DBI ↓
TNC	**0.71±0.01**	**0.36±0.01**	**0.44±0.02**	**0.74±0.04**	**0.61±0.02**	**0.52±0.04**
CPC	0.51±0.03	0.84±0.06	0.26±0.02	1.44±0.04	0.58±0.02	0.57±0.05
T-Loss	0.61±0.08	0.64±0.12	0.25±0.01	1.30±0.03	0.17±0.01	1.76±0.20
K-means	0.01±0.019	7.23±0.14	0.19±0.11	3.65±0.48	0.12±0.40	2.66±0.05

Table 1: Clustering quality of representations in the encoding space.

Table 1 summarizes the scores for all baselines and across all datasets, demonstrating that TNC is superior in learning representations that can distinguish the latent dynamics of time series. CPC performs closely to T-loss on waveform data, but performs poorly on the simulated dataset, where signals are highly non-stationary and transitions are less predictable. For the HAR dataset however, CPC clusters states very well, because most activities are recorded in a specific order, empowering predictive coding. T-loss performs reasonably well in simulated setting, however it fails to distinguish states 0 and 2, where signals come from autoregressive models with different parameters and thus have a rather similar generative process. Performing K-means on original time series doesn't generate coherent cluster, as demonstrated by the scores. The performance is slightly better in time series like the ECG waveforms, where the signals are formed by consistent shapelets, therefore the DTW measures similarity more accurately.

4.2 Evaluation: Classification

We further evaluate the quality of the encodings using a classification task. We train a linear classifier to evaluate how well the representations can be used to classify hidden states. Performance is compared to a supervised classifier, composed of an encoder and a classifier with identical architectures to that of the unsupervised models, as well as a K-nearest neighbor classifier using DTW metric. Performance is reported as the prediction accuracy as well as the area under the precision-recall curve (AUPRC) score, since AUPRC is a more accurate reflection of model performance for imbalance classification settings like the waveform dataset.

	Simulation		ECG Waveform		HAR	
Method	AUPRC	Accuracy	AUPRC	Accuracy	AUPRC	Accuracy
TNC	**0.99±0.00**	**97.52±0.13**	**0.52±0.03**	**77.79±0.84**	**0.85±0.02**	**89.41±0.81**
CPC	0.69±0.06	70.26±6.48	0.35±0.03	68.64±0.49	0.83±0.01	88.68±1.62
T-Loss	0.78±0.01	76.66±1.40	0.45±0.01	75.51±1.26	0.63±0.00	70.06±4.64
KNN	0.42±0.00	55.53±0.65	0.38±0.06	54.76±5.46	0.75±0.01	84.85±0.84
Supervised	**0.99±0.00**	**98.56±0.13**	**0.64±0.01**	**94.81±0.28**	**0.98±0.00**	**95.39±0.1**

Table 2: Performance of all baselines in classifying the underlying hidden states of the time series, measured by accuracy and AUPRC score.

Table 2 demonstrates the classification performance for all datasets. The performance of classifiers that use TNC representations is closer to the end-to-end supervised model, in comparison to CPC and Triplet Loss. This provides further evidence that our encodings capture informative parts of the time series. In datasets like the HAR, where an inherent ordering usually exists, CPC performs reasonably. However, in datasets with increased non-stationarity, the performance drops. Triplet-loss is also a powerful framework, but since it samples positive examples from overlapping windows of time series, it is vulnerable to only mapping the overlaps into the encoding and therefore failing to learn more general representations. TNC on the other hand, samples similar windows from a wider distribution, defined by the temporal neighborhood, where many of the neighboring signals don't necessarily overlap. Lower performance of CPC and Triplet-Loss can also be partly due to the fact that none of these methods explicitly account for the potential sampling bias that happens when randomly selected negative examples are similar to the reference W_t.

4.3 Evaluation: Trajectory

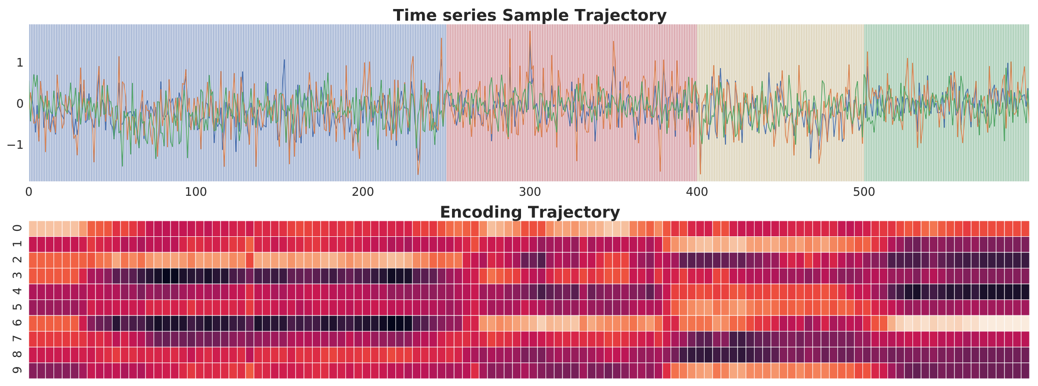


Figure 3: Trajectory of a signal encoding. The top plot shows the original time series with shaded regions indicating the underlying state. The bottom plot shows the 10 dimensional encoding of the sliding windows W_t where $\delta = 50$.

In this section we investigate the trajectories of our learnt encodings over time, to understand how the state transition is modeled in the representation space. Figure 3 shows a sample from the simulated dataset over time. The top panel shows the signal measurements and the shaded regions indicate the underlying latent states. The bottom panel illustrates the 10-dimensional representation of a sliding window W_t estimated over time. From the bottom panel of Figure 3, we can see that the encoding pattern changes at state transitions and settle into a different pattern, corresponding to the new state. This happens at every transition, and we can see the distinct patterns for all 4 underlying states in the representations. This analysis of the trajectory of change could be very informative for the post analysis by the users; for instance in clinical applications, it could help clinicians visualize the evolution of the patient state over time and plan treatment based on the state progression.

5 Related work

While integral for many applications, unsupervised representation learning has been far less studied in the time series (Längkvist et al., 2014), compared to other domains such as vision or natural language processing (Denton et al., 2017; Radford et al., 2015; Gutmann & Hyvärinen, 2012; Wang & Gupta, 2015). One of the earliest approaches to unsupervised end-to-end representation learning in time series is the use of auto-encoders (Choi et al., 2016a; Amiriparian et al., 2017; Malhotra et al., 2017) and seq-to-seq models (Lyu et al., 2018), with the objective to train an encoder jointly with a decoder that reconstructs the input signal from its learned representation. Using fully generative models like variational auto-encoders is also useful for imposing properties like disentanglement, that help with the interpretability of the representations (Dezfouli et al., 2019). However, in many cases, like for high-frequency physiological signals, the reconstruction of complex time series can be challenging; therefore, more novel approaches are designed to avoid this step. Contrastive Predictive Coding (Oord et al., 2018; Löwe et al., 2019) learns representations by predicting the future in latent space, eliminating the need to reconstruct the full input. The representations are such that the mutual information between the original signal and the concept vector is maximally preserve using a lower bound approximation and a contrastive loss. Very similarly, in Time Contrastive Learning (Hyvarinen & Morioka, 2016), a contrastive loss is used to predict the segment-ID of multivariate time-series as a way to extract representation. Franceschi et al. (2019) employs time-based negative sampling and a triplet loss to learn scalable representations for multivariate time series. Some approaches use inherent similarities in temporal data to learn representations without supervision. For instance, in similarity-preserving representation learning (Lei et al., 2017), learned encodings are constrained to preserve the pairwise similarities that exist in the time domain, measured by DTW distance. Another group of approaches combines reconstruction loss with clustering objectives in order to cluster similar temporal patterns in the encoding space (Ma et al., 2019; Madiraju et al., 2018).

In healthcare, learning representation of rich temporal medical data is extremely important for understanding patients' underlying health conditions. However, most of the existing approaches for learning representations are designed for specific downstream tasks and require labeling by experts (Choi et al., 2016b;c). Some examples of similar works to representation learning in the field of clinical ML include computational phenotyping for discovering subgroups of patients with similar underlying disease mechanisms from temporal clinical data (Lasko et al., 2013; Suresh et al., 2018; Schulam et al., 2015). Also, disease progression modeling, for learning the hidden vector of comorbidities representing a disease over time (Wang et al., 2014; Alaa & van der Schaar, 2018).

6 Conclusion

In this paper we present a novel unsupervised representation learning framework for complex multivariate time series, called Temporal Neighborhood Coding (TNC). This framework is designed to learn the underlying dynamics of non-stationary signals and to model the progression over time, by defining a temporal neighborhood. The problem is motivated from the medical field where patients transition between distinct clinical states over time, and obtaining labels to define these underlying states is challenging. We evaluate the performance of TNC on multiple datasets and show that our representations are generalizable and can easily be used for diverse tasks such as classification and clustering. We finally note that TNC is flexible to be used with arbitrary encoder architectures, therefore the framework is applicable to many time series data domains. Moreover, in addition to tasks presented in this paper, general representations can be used for a number of other downstream task such as anomaly detection, which is challenging in supervised learning settings for time series data in sparsely labeled contexts.

REFERENCES

Ahmed M Alaa and Mihaela van der Schaar. Forecasting individualized disease trajectories using interpretable deep learning. *arXiv preprint arXiv:1810.10489*, 2018.

Shahin Amiriparian, Michael Freitag, Nicholas Cummins, and Björn Schuller. Sequence to sequence autoencoders for unsupervised representation learning from audio. In *Proc. of the DCASE 2017 Workshop*, 2017.

Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Edward Choi, Mohammad Taha Bahadori, Elizabeth Searles, Catherine Coffey, Michael Thompson, James Bost, Javier Tejedor-Sojo, and Jimeng Sun. Multi-layer representation learning for medical concepts. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1495–1504, 2016a.

Edward Choi, Andy Schuetz, Walter F Stewart, and Jimeng Sun. Medical concept representation learning from electronic health records and its application on heart failure prediction. *arXiv preprint arXiv:1602.03686*, 2016b.

Youngduck Choi, Chill Yi-I Chiu, and David Sontag. Learning low-dimensional representations of medical concepts. *AMIA Summits on Translational Science Proceedings*, 2016:41, 2016c.

Ching-Yao Chuang, Joshua Robinson, Lin Yen-Chen, Antonio Torralba, and Stefanie Jegelka. Debiased contrastive learning. *arXiv preprint arXiv:2007.00224*, 2020.

Emily L Denton et al. Unsupervised learning of disentangled representations from video. In *Advances in Neural Information Processing Systems*, pp. 4414–4423, 2017.

Amir Dezfouli, Hassan Ashtiani, Omar Ghattas, Richard Nock, Peter Dayan, and Cheng Soon Ong. Disentangled behavioural representations. In *Advances in Neural Information Processing Systems*, pp. 2251–2260, 2019.

Jeff Donahue and Karen Simonyan. Large scale adversarial representation learning. In *Advances in Neural Information Processing Systems*, pp. 10541–10551, 2019.

Marthinus C Du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems*, pp. 703–711, 2014.

Marthinus Christoffel Du Plessis and Masashi Sugiyama. Class prior estimation from positive and unlabeled data. *IEICE TRANSACTIONS on Information and Systems*, 97(5):1358–1362, 2014.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 213–220, 2008.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963, 2019.

Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. Unsupervised scalable representation learning for multivariate time series. In *Advances in Neural Information Processing Systems*, pp. 4652–4663, 2019.

Michael U Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research*, 13 (Feb):307–361, 2012.

Aapo Hyvarinen and Hiroshi Morioka. Unsupervised feature extraction by time-contrastive learning and nonlinear ICA. In *Advances in Neural Information Processing Systems*, pp. 3765–3773, 2016.

Ryuichi Kiryo, Gang Niu, Marthinus C Du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems*, pp. 1675–1685, 2017.

Martin Längkvist, Lars Karlsson, and Amy Loutfi. A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, 42:11–24, 2014.

Thomas A Lasko, Joshua C Denny, and Mia A Levy. Computational phenotype discovery using unsupervised feature learning over noisy, sparse, and irregular clinical data. *PloS one*, 8(6), 2013.

Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *International Conference on Machine Learning*, volume 3, pp. 448–455, 2003.

Qi Lei, Jinfeng Yi, Roman Vaculin, Lingfei Wu, and Inderjit S Dhillon. Similarity preserving representation learning for time series clustering. *arXiv preprint arXiv:1702.03584*, 2017.

Xiaoli Li and Bing Liu. Learning to classify texts using positive and unlabeled data. In *IJCAI*, volume 3, pp. 587–592, 2003.

Sindy Löwe, Peter O'Connor, and Bastiaan Veeling. Putting an end to end-to-end: Gradient-isolated learning of representations. In *Advances in Neural Information Processing Systems*, pp. 3039–3051, 2019.

Xinrui Lyu, Matthias Hueser, Stephanie L Hyland, George Zerveas, and Gunnar Rätsch. Improving clinical predictions through unsupervised time series representation learning. *arXiv preprint arXiv:1812.00490*, 2018.

Qianli Ma, Jiawei Zheng, Sen Li, and Gary W Cottrell. Learning representations for time series clustering. In *Advances in Neural Information Processing Systems*, pp. 3776–3786, 2019.

Naveen Sai Madiraju, Seid M Sadat, Dimitry Fisher, and Homa Karimabadi. Deep temporal clustering: Fully unsupervised learning of time-domain features. *arXiv preprint arXiv:1802.01059*, 2018.

Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. Timenet: Pre-trained deep recurrent neural network for time series classification. *arXiv preprint arXiv:1706.08838*, 2017.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

George Moody. A new method for detecting atrial fibrillation using RR intervals. *Computers in Cardiology*, pp. 227–230, 1983.

Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Slobodan Petrovic. A comparison between the silhouette index and the davies-bouldin index in labelling ids clusters. In *Proceedings of the 11th Nordic Workshop of Secure IT Systems*, pp. 53–64, 2006.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. Learning to generate reviews and discovering sentiment. *arXiv preprint arXiv:1704.01444*, 2017.

Peter Schulam, Fredrick Wigley, and Suchi Saria. Clustering longitudinal clinical marker trajectories from electronic health data: Applications to phenotyping and endotype discovery. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

Harini Suresh, Jen J Gong, and John V Guttag. Learning tasks for multitask learning: Heterogenous patient populations in the icu. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 802–810, 2018.

Xiang Wang, David Sontag, and Fei Wang. Unsupervised learning of disease progression models. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 85–94, 2014.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2794–2802, 2015.

Qiang Yang and Xindong Wu. 10 challenging problems in data mining research. *International Journal of Information Technology & Decision Making*, 5(04):597–604, 2006.

Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *IEEE Computational Intelligence Magazine*, 13(3):55–75, 2018.

A Appendix

A.1 Simulated Dataset

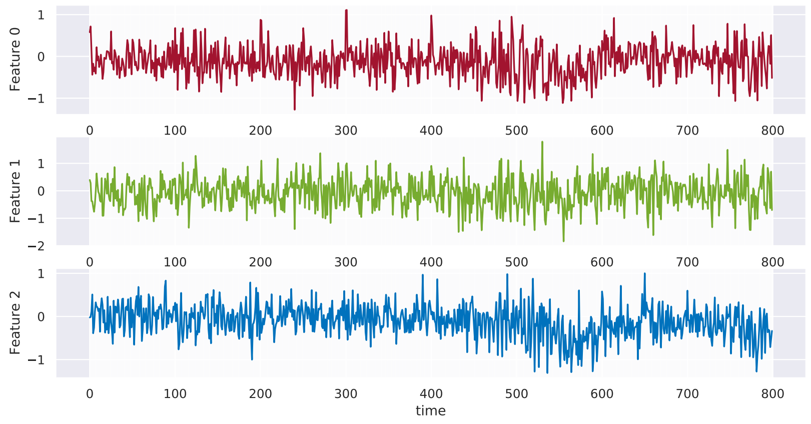


Figure A.1: A normalized time series sample from the simulated dataset for a single individual. Each row represents a single feature. The shaded regions indicate one of the 4 simulated states.

The simulated time series consists of 3 features, generated from different underlying hidden states. Figure A.1 shows a sample from this dataset. Each panel in the figure shows one of the features and the shaded regions indicate the underlying state of the signal in that time span. We use a hidden Markov Model to generate this random latent states over time. The transition probability is set equally to %5 for switching to an all alternating state, and %85 for not changing state. In each state, the time series are generated from a different signal distribution. Table 3 describes the distribution of each signal feature in each state. Note that feature 1 and 2 are always correlated, mainly to resemble realistic clinical time series, for instance, physiological measurements like pulse rate and heart rate are always correlated.

	State 1	State 2	State 3	State 4
Feature 1	GP (periodic)	NARMA$_\alpha$	GP (Squared Exp.)	NARMA$_\beta$
Feature 2	GP (periodic)	NARMA$_\alpha$	GP (Squared Exp.)	NARMA$_\beta$
Feature 3	GP (Squared Exp.)	NARMA$_\beta$	GP (periodic)	NARMA$_\alpha$

Table 3: Signal distributions for each time series feature of the simulated dataset

For instance, in state 1, the correlated features are generated by a Gaussian Process (GP) with a periodic kernel. Feature 3, that is uncorrelated with the other 2 features comes from another GP with squared exponential kernel. In addition to GPs, we also have multiple Non-Linear Auto-Regressive Moving Average (NARMA) time series models. The linear function of NARMA$_\alpha$ and NARMA$_\beta$ are shown in Equation 2 and 3.

$$\text{NARMA}_\alpha : y(k+1) = 0.3y(k) + 0.05y(k)\sum_{i=0}^{n-1} y(k-i) + 1.5u(k-(n-1))u(k) + 0.1 \quad (2)$$

$$\text{NARMA}_\beta : y(k+1) = 0.1y(k) + 0.25y(k)\sum_{i=0}^{n-1} y(k-i) + 2.5u(k-(n-1))u(k) - 0.005 \quad (3)$$

A white Gaussian noise with $\sigma = 0.3$ is added to all signals, and overall, the dataset consists of 500 instances of $T = 2000$ measurements.

A.2 Baseline implementation details

Implementation of all baselines are included in the code base for reproducability purposes, and hyper-parameters for all baselines are tuned using cross-validation.

Contrastive Predictive Coding (CPC): The CPC baseline first processes the sequential signal windows using a encoder $Z_t = Enc(X_t)$, with a similar architecture to the encoders of other baselines. Next, an autoregressive model g_{ar} aggregates all the information in $Z_{\leq t}$ and summarizes it into a context latent representation $c_t = g_{ar}(Z_{\leq t})$. In our implementation we have used a single layer, one-directional recurrent neural network with GRU cell and hidden size equal to the encoding size as the auto-regressor. Similar to the original paper, the density ratio is estimated using a linear transformation and the model is trained for 1 step ahead prediction.

Triplet-Loss (T-Loss): The triplet loss baseline is implemented using the original code made available by the authors on Github[2].

KNN and K-means: These two baselines for classification and clustering are implemented using the tslearn library [3], that integrates distance metrics such as DTW. Note, evaluating DTW is computationaly expensive and the tslearn implementation is not optimized, therefore for the waveform data with windows of size 2500, we had to down-sampled the signal frequency by a factor of 2.

A.3 TNC implementation extra details

In order to define the nighborhood range in the TNC framework, as mentioned earlier, we use the Augmented-Dickey Fuller (ADF) statistical test, to determine this range (η) as the region for which the signals remains stationary. More precisely, we gradually increase the range, from a single window size up to 3 times the window size (our upper limit), and repeatedly perform the ADF test. We use the p-value from this statistical test to determine whether the Null hypothesis can be rejected, meaning that the signal is stationary. At the point where the p-value is above our defined threshold (0.01), we can no longer assume that the signal is stationary and this is where we set the η parameter. Now, once the Neighborhood is defined, we make sure the non-nieghboring samples are taken from the distribution of windows with at least $4 \times \eta$ away from W_t, ensuring low likelihood of belonging to the neighborhood. Note that for implementation of ADF, we use the stats model library [4]. Unfortunately, this implementation is not optimized and doesn't support GPU implementation, therefore evaluating the neighborhood range using ADF slows down the training of TNC framework. As a future direction, we would like to work on a more efficient implementation of the ADF score for our framework.

A.4 Clustering metrics

Most cluster validity measures assess certain structural properties of a clustering result. In our evaluation we have used 2 measures, namely the Silhouette score and Davies-Bouldin index, to evaluate the clustering quality of the representations. Davies-Bouldin measures intra-cluster similarity (coherence) and inter-cluster differences (separation). Let $\mathcal{C} = \{\mathcal{C}_1, ..., \mathcal{C}_k\}$ be a clustering of a set D of objects. Davies-Bouldin score can is evaluated as follows:

$$DB = \frac{1}{k}\sum_{i=1}^{k} max_j \frac{s(\mathcal{C}) + s(\mathcal{C})}{\delta \mathcal{C}_i \mathcal{C}_j} \tag{4}$$

where $s(\mathcal{C})$ measures the scatter within a cluster, and δ is a cluster to cluster distance measure.

The silhouette score on the other hand, measures how similar an object is to its own cluster *compared* to other clusters. Both measures are commonly used for evaluation of clustering algorithm. A comparison of 2 metric has shown that in some cases, Silhouette index produces slightly more accurate results, however, Davies-Bouldin index is much less complex in terms of computation Petrovic (2006).

[2]https://github.com/White-Link/UnsupervisedScalableRepresentationLearningTimeSeries

[3]https://tslearn.readthedocs.io/en/stable/index.html

[4]https://www.statsmodels.org/dev/generated/statsmodels.tsa.stattools.adfuller.html

A.5 Setting the weights for PU learning

As mentioned in the Experiment section, in order to determine the weight parameter for debiasing the loss, we evaluate TNC loss for different weights. Table 4 shows these values for the different experiment settings. The loss column reports the value measured in Equation 1, and the accuracy shows how well the discriminator identifies the neighboring samples from non-neighboring ones. We pick the optimal weight for each setting, as the one that minimized TNC objective loss.

	Simulation		ECG Waveform		HAR	
Weight	Loss	Accuracy	Loss	Accuracy	Loss	Accuracy
0.2	0.582±0.002	74.29±0.61	0.631±0.011	60.44±2.56	0.475±0.004	85.75±0.5
0.1	**0.571±0.011**	75.41±0.37	0.637±0.011	63.67±1.29	0.413±0.003	88.21±1.29
0.05	0.576±0.002	75.73±0.24	**0.622±0.023**	66.04±3.46	**0.383±0.001**	87.33±0.17

Table 4: Training the TNC framework using different weight parameters. The loss is the measured value determined in Equation 1, and the Accuracy is the accuracy of the discriminator.

In addition, in order to assess the impact of re-weighting the loss on downstream classification performance, we compared these performance measures for weighted and non-weighted settings. Table 5 demonstrates these results and confirms that weight adjusting the loss for non-neighboring samples, improves the quality of learnt representations.

Weighting?	Simulation	ECG Waveform	HAR
True	**97.52±0.13**	**77.79±0.84**	**89.41±0.81**
False	97.17±0.44	75.26±1.48	75.25±13.6

Table 5: Classification accuracy on downstream tasks with the TNC frameworks, using 2 different weighting strategies: 1)Trained with weight adjustment, 2)Trained with $w = 0$.

A.6 Supplementary Figure

A.6.1 HAR data

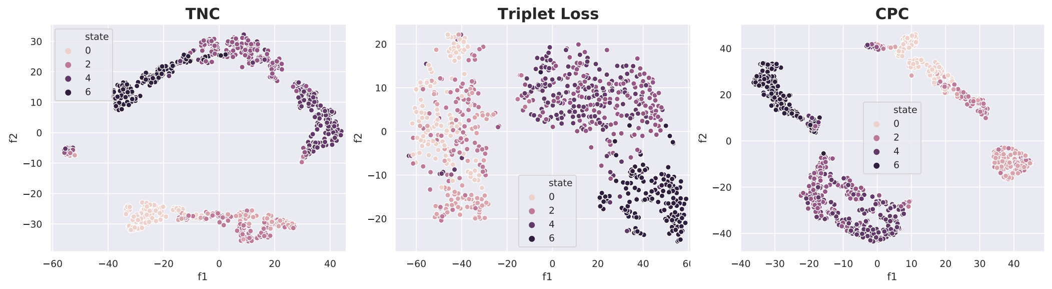


Figure A.2: T-SNE visualization of **HAR** signal representations for all baselines. Each point in the plot is a 10 dimensional representation of a window of $\delta = 4$, with colors indicating latent states.

A.6.2 Clinical waveform data

In order to understand what TNC framework learns from the high dimensional ECG signals, we visualize the trajectory of the representations of an individual sample over time. Figure A.5 demonstrates this example, where the top 2 rows are ECG signals from two recording leads and the bottom row demonstrates the representation vectors. We see that around second 40 the pattern in the representations change as a result of an artifact happening in one of the signals. Also, with the help of our clinical expert, we tried to interpret different patterns in the encoding space. For instance, between time 80 and 130, where features 0-10 become more activated, the heart rate (HR) increases

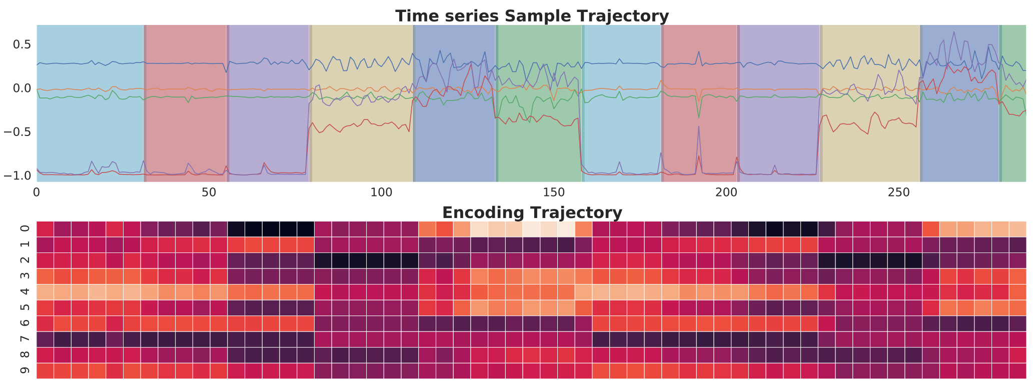


Figure A.3: Trajectory of a **HAR** signal encoding. The top plot shows the original time series with shaded regions indicating the underlying state. The bottom plot shows the 10 dimensional encoding of the sliding windows W_t where $\delta = 4$.

has increased. Increase in HR can be seen as increased frequency in the ECG signals and is one of the indicators of arrhythmia that we believe TNC.

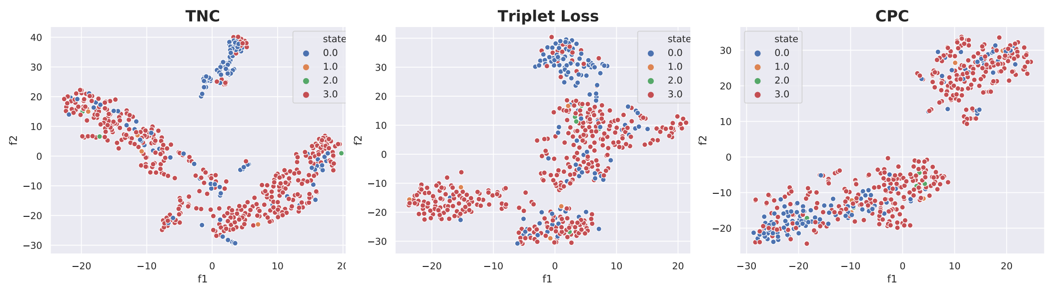


Figure A.4: T-SNE visualization of **waveform** signal representations for unsupervised representation learning baselines. Each point in the plot is a 64 dimensional representation of a window of time series, with the color indicating the latent state.

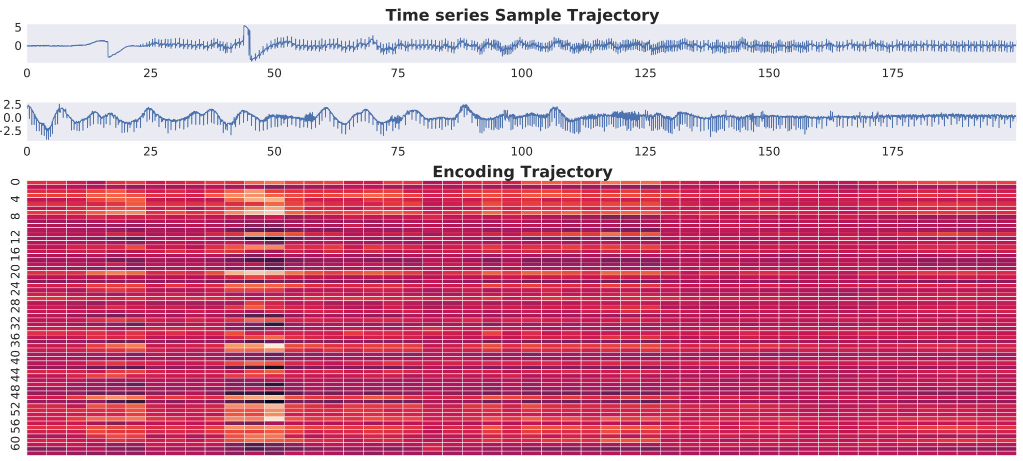


Figure A.5: Trajectory of a **waveform** signal encoding. The top two plot shows the ECG recordings from 2 ECG leads. The bottom plot shows the 64 dimensional encoding of the sliding windows W_t where $\delta = 2500$.