The AAAI 2023 Workshop on Representation Learning for Responsible Human-Centric AI (R^2HCAI)

Temporal Supervised Contrastive Learning with Applications to Tabular Time Series Data

Shahriar Noroozizadeh[1] **Jeremy C. Weiss**[2] **George H. Chen**[1]

[1] Carnegie Mellon University [2] National Library of Medicine
snoroozi@andrew.cmu.edu, jeremy.weiss@nih.gov, georgechen@cmu.edu

Abstract

We propose a contrastive learning framework for time series data, where each time step has its own classification label and is mapped to a point in an embedding space. This framework has the following properties: (1) nearby points in the embedding space have similar predicted class probabilities, (2) adjacent time steps of the same time series map to nearby points in the embedding space, and (3) time steps with very different raw features map to far apart regions of the embedding space *even if these two time steps have the same classification label*. Whereas properties (1) and (2) are standard, including property (3) is novel and is achieved using a *nearest neighbor pairing* mechanism in the raw feature space. We demonstrate our framework on tabular time series data: on a synthetic dataset with ground truth embedding space structure, our method consistently recovers the structure across experimental repeats whereas baselines do not; on two real clinical datasets, our method achieves competitive accuracy scores. Ablation studies reveal the importance of including the nearest neighbor pairing mechanism.

1 Introduction

In recent years, many advances in representation learning have been in the paradigm of *contrastive learning* (e.g., Oord, Li, and Vinyals (2018); Chen et al. (2020a,b)), which aims to learn an embedding representation of raw data where "similar" data points have embeddings that are close by to each other whereas "dissimilar" data points have embeddings that are far apart. The goal is to have the embedding space capture interesting problem-specific semantic structure. While contrastive learning has typically been stated in the unsupervised setting, recently Khosla et al. extended it to the classification setting to obtain *supervised contrastive learning* (SCL) (Khosla et al. 2020).

In this paper, we extend SCL to incorporate temporal dynamics, yielding a framework we call TEMPORAL-SCL. Our framework learns an embedding representation of time series at the individual time step level (i.e., each time step is mapped to an embedding vector) and has the following properties:

- (Predictive) The time-dependent embeddings are helpful in predicting a classification outcome, where we consider both the *static* outcome case where a single time series is associated with a single class, and the *dynamic* outcome case where a single time series has time-varying classification label.
- (Temporally smooth) When we look at the embeddings of two adjacent time steps from the same time series, these embeddings tend to be close by.
- (Diverse in capturing raw feature heterogeneity) Embeddings that correspond to very different raw inputs tend to be mapped to different parts of the embedding space *even if they have the same classification outcome*.

The first property is standard and says that making a classifier that uses the embeddings to make predictions should work well. This prediction is for a single time step and corresponds to how prediction is done using self-supervised (Chen et al. 2020a) or supervised contrastive learning (Khosla et al. 2020). The second property has already appeared in self-supervised constrastive learning that accounts for temporal dynamics (Dave et al. 2022) as well as other non-contrastive learning temporal models (e.g., Lee, Yoon, and Van Der Schaar 2019). However, to the best of our knowledge, no existing contrastive learning approach yields time-dependent embeddings that have all three of the above properties.

The third property (diverse in capturing raw feature heterogeneity) is more nuanced. It says that parts of the embedding space that are far apart could potentially still be associated with the same classification outcome. In contrast, if we only care about the first two properties and not this last property, then the framework would not penalize a learned embedding space in which all raw inputs of the same class—regardless of whether they are actually close to each other in the original raw feature space—get mapped to approximately the same embedding space location. Although such a learned embedding space could be very helpful for classification, it is not designed to distinguish between the different patterns of raw inputs that lead to the same predicted class.

As a running example used throughout this paper, consider a clinical application where the different time series come from different patients, and each time series consists of clinical features recorded over time (e.g., blood pressure, temperature). Different time series could vary in length and the time steps could be sampled irregularly (the time elapsed between consecutive time steps need not be the same). In the static outcome case, we aim to predict a patient's outcome (e.g., in-hospital mortality) at the final time step of

the patient's time series. In the dynamic outcome case, we aim to predict how a disease state for a patient changes over time (e.g., progression of Alzheimer's discretized into a few stages). Importantly, discovering different patient measurements that are indicative of the same outcome could be helpful. The reason for this is that even though two patients may have the same predicted outcome, how a clinician plans treatment for them could differ if the two patients have very different patient characteristics. For this sort of tabular clinical data, there is no standard data augmentation procedure commonly accepted as being sufficiently realistic for clinical use. For this reason, we describe our contrastive learning framework *without data augmentation*. For applications in which data augmentation makes sense, adding data augmentation to our framework is straightforward, as we point out later.

Concretely, our main contributions are as follows:

- We propose TEMPORAL-SCL, which learns embeddings that are predictive, temporally smooth, and diverse in capturing raw feature heterogeneity. This last property is achieved by including a *nearest neighbor pairing* mechanism.
- To probe whether the learned embedding space captures any sort of interesting phenomena in terms of raw features, we present a clustering-based heatmap visualization of TEMPORAL-SCL embeddings, relating the embedding space to both raw features and to prediction outcomes. Note that the clustering is done *after* and not as part of learning a TEMPORAL-SCL model as to avoid imposing strong assumptions such as there being a specific number of clusters.
- In a synthetic dataset with known ground truth embedding space structure, we show that TEMPORAL-SCL correctly recovers the underlying structure (100% of the time across 10 experimental repeats with different random seeds). Removing the nearest neighbor pairing mechanism results in 0% recovery. Meanwhile, all baselines fail to recover the correct structure the majority of the time (across 10 experiment repeats per baseline).
- In two real clinical datasets, we show that TEMPORAL-SCL achieves competitive accuracy scores compared to various baselines. We also show that removing the nearest neighbor pairing mechanism results in accuracy scores that either stay about the same or are worse, while resulting in significantly worse scores on *unsupervised* metrics.

2 Background

We state the time series prediction setup we consider (Section 2.1) and then review supervised contrastive learning (Khosla et al. 2020) (Section 2.2). Throughout the paper, for any positive integer m, we denote $[m] := \{1, 2, \ldots, m\}$.

2.1 Problem Setup

Training data. We assume that the training dataset consists of N different time series. We view each time series as a single data point. For the i-th data point (with $i \in [N]$), we observe L_i time steps, where at each time step, we keep track of a total of D features. Specifically, we denote the i-th data point's feature vector at time step $\ell \in [L_i]$ (sorted chronologically) as $\mathbf{x}_i^{(\ell)} \in \mathbb{R}^D$. Moreover, we also know the actual times that the time steps correspond to, where we denote the i-th data point's time at the ℓ-th time step as $t_i^{(\ell)} \in \mathbb{R}$. This notation allows for some features to be static, i.e., a feature could stay constant across time.

We further assume that we have ground truth information. Specifically we assume that every time step of a time series belongs to one of C different classes, i.e., the set of classes is $[C]$. In the dynamic outcome case, where the classification label varies over time, we assume that we know the classification label $y_i^{(\ell)} \in [C]$ for every data point $i \in [N]$ for every time step $\ell \in [L_i]$. For the static outcome case, we only know the classification label at the final time step per time series, i.e., we know $y_i^{(L_i)} \in [C]$ for all $i \in [N]$. At earlier time steps, we set $y_i^{(\ell)} =$ "?" for all $\ell < L_i$, which could be thought of as an additional "unknown" class.

Prediction task. Given a test time series, suppose that we observe the feature vector $\mathbf{x}_* \in \mathbb{R}^D$ at a *single time step*. We aim to predict the target label $y_* \in [C]$ corresponding to $\mathbf{x}_*$. In the dynamic outcome setting, this target label is for the same time step as $\mathbf{x}_*$. In the static outcome setting, we take the target label to be the label for the final time step in the test time series.

We focus on this setup of predicting the classification output for a single time step's worth of information for simplicity and also because it captures the extreme case of what is possible: that a test time series just consists of one time step. In a clinical setup, this could happen if a patient enters a new hospital for which we see measurements of this patient for the first time; the patient could have previously been at a different hospital that we do not have data from.

2.2 Supervised Contrastive Learning (SCL)

SCL learns an embedding representation of the data so that "similar" data points have embeddings that have higher cosine similarity compared to those of "dissimilar" data points. Data points are "similar" if they have the same classification label. As this framework was originally developed without temporal structure, we drop superscripts previously used to indicate dependence on time. For ease of exposition, we present a simplified version of SCL that we call SIMPLE-SCL, *which does not use data augmentation* (since our experiments later will be on tabular data without data augmentation).

Notation. We use f to denote the so-called *encoder network*, where given any point $\mathbf{x}$ in the raw input feature space, its embedding representation is $f(\mathbf{x})$. This embedding representation is constrained to be a d-dimensional Euclidean vector with norm 1, also referred to as a *hyperspherical embedding*. Thus, $\|f(\mathbf{x})\| = 1$. We denote this hyperspherical output space as $\mathcal{S}^{d-1} := \{\mathbf{z} \in \mathbb{R}^d \text{ s.t. } \|\mathbf{z}\| = 1\}$. For $\mathbf{u}, \mathbf{v} \in \mathcal{S}^{d-1}$, the cosine similarity between $\mathbf{u}$ and $\mathbf{v}$ is $\frac{\langle \mathbf{u}, \mathbf{v} \rangle}{\|\mathbf{u}\|\|\mathbf{v}\|} = \langle \mathbf{u}, \mathbf{v} \rangle$, where $\langle \cdot, \cdot \rangle$ denotes the Euclidean dot product.

SIMPLE-SCL. We learn the encoder f using minibatch gradient descent. For a minibatch of B training

points $\mathbf{x}_1, \ldots, \mathbf{x}_B$ with corresponding classification labels $y_1, \ldots, y_B$, we denote the embeddings of these points as $\mathbf{z}_1 = f(\mathbf{x}_1), \ldots, \mathbf{z}_B = f(\mathbf{x}_B)$. Since we want points with the same label to have high cosine similarity, we keep track of which points have the same label. To do this, we let $\mathcal{P}(i)$ denote the set of points with the same label as the i-th point, excluding the i-th point:

$$\mathcal{P}(i) := \{j \in [B] \text{ s.t. } y_j = y_i \text{ and } j \neq i\}. \quad (1)$$

Next, we define the following ratio:

$$\Psi(i, j; \tau) := \frac{\exp(\langle \mathbf{z}_i, \mathbf{z}_j \rangle / \tau)}{\sum_{k \in [B] \text{ s.t. } k \neq i} \exp(\langle \mathbf{z}_i, \mathbf{z}_k \rangle / \tau)}, \quad (2)$$

where the constant $\tau > 0$ is a user-specified hyperparameter. The key idea is that if the i-th and j-th points have the same label (i.e., $j \in \mathcal{P}(i)$), then we want $\Psi(i, j; \tau)$ to be large: the numerator being large means that cosine similarity $\langle \mathbf{z}_i, \mathbf{z}_j \rangle$ is large while the denominator provides a normalization to ensure that $\Psi(i, j; \tau) \in [0, 1]$. Then to encourage $\Psi(i, j; \tau)$ to be large for all $i \in [B]$ and $j \in \mathcal{P}(i)$, we can minimize the loss

$$L_{\text{Simple-SCL}} := - \sum_{i \in [B] \text{ s.t. } |\mathcal{P}(i)| \geq 1} \frac{1}{|\mathcal{P}(i)|} \sum_{j \in \mathcal{P}(i)} \log \Psi(i, j; \tau). \quad (3)$$

The original SCL uses a loss of the same form but includes data augmentation (see Appendix for details).

3 Method

We now present our proposed temporal variant of SCL called TEMPORAL-SCL, which consists of three networks: an encoder network f (maps a specific time step's feature vector to an embedding), a predictor network g (maps an embedding to predicted class probabilities), and a temporal network h (for encouraging temporal smoothness). We first give an overview of these three networks (Section 3.1) prior to describing how they are trained (Section 3.2). We then describe a clustering-based method for visualizing learned embeddings (Section 3.5).

3.1 Overview of Encoder, Predictor, and Temporal Networks

Encoder network f. We aim to learn an embedding representation of every time step of the different training time series. Just as in Section 2.2, this amounts to learning an encoder network f that maps from the raw feature vector space (now for just a single time step) to the hyperspherical space $\mathcal{S}^{d-1}$. Since we now account for time steps, we add superscripts: we let $\mathbf{z}_i^{(\ell)} := f(\mathbf{x}_i^{(\ell)})$ denote the embedding of the i-th data point's feature vector at time step ℓ.

Aside from using time steps, the major difference between TEMPORAL-SCL and SIMPLE-SCL is that for TEMPORAL-SCL, feature vectors are considered similar if they simultaneously have similar outcomes (i.e., classification labels $y_i^{(\ell)}$) *and* similar input feature vectors (unlike with SIMPLE-SCL, where similarity is purely based on classification labels).

Predictor network g. As we want the learned embedding space to be predictive of classification outcomes, we learn a predictor network g that maps from the embedding space $\mathcal{S}^{d-1}$ to the space of probability distributions over C classes.

Temporal network h. To encourage two adjacent time steps in the same time series $\mathbf{x}_i^{(\ell)}$ and $\mathbf{x}_i^{(\ell+1)}$ to map to embeddings $\mathbf{z}_i^{(\ell)}$ and $\mathbf{z}_i^{(\ell+1)}$ that are close to each other, we learn a temporal network h that predicts how embeddings change over time. For the i-th training data point, we define the time duration $\delta_i^{(\ell)} := t_i^{(\ell+1)} - t_i^{(\ell)}$ for $\ell \in [L_i - 1]$. Then h takes as input the sequence $(\mathbf{z}_i^{(1)}, \delta_i^{(1)}), \ldots, (\mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)})$ and outputs a prediction for $\mathbf{z}_i^{(\ell+1)}$. We ask that $\mathbf{z}_i^{(\ell+1)}$ and $h\big((\mathbf{z}_i^{(1)}, \delta_i^{(1)}), \ldots, (\mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)})\big)$ be close (using squared Euclidean distance loss). Thus, h aims to make the next time step's embedding predictable based on all previous time steps and could be thought of as a regularization term. A similar temporal smoothness regularization strategy was used by Lee, Yoon, and Van Der Schaar (2019).

3.2 Overview of TEMPORAL-SCL Training and Prediction

Training. We train the three networks of TEMPORAL-SCL in three phases:

1. (Pre-training) We initialize the hyperspherical embedding space by pre-training the encoder network f using data at the individual time step level and SIMPLE-SCL with cosine-similarity. Note that during this phase, we do not model nor use temporal structure, and we effectively treat the different time steps as separate. We explain this phase in more detail in Section 3.3, and our experiments later show that this phase significantly improves the model's prediction accuracy.
2. (Joint optimization of encoder and temporal networks) After pre-training the encoder f, we then jointly optimize f and the temporal network h. This phase accounts for temporal structure. Details of this phase are in Section 3.4.
3. (Learning the predictor network) At this point, we treat the encoder f as fixed, so we can compute all the training embeddings at different time steps (the $\mathbf{z}_i^{(\ell)}$ variables). In the dynamic outcome case, we learn the predictor network g by treating the $\mathbf{z}_i^{(\ell)}$ variables as input feature vectors and the corresponding $y_i^{(\ell)}$ variables as target labels, minimizing cross-entropy loss. In the static outcome case, we instead set the target label for $\mathbf{z}_i^{(\ell)}$ to be the final time step's label $y_i^{(L_i)}$. As this phase amounts to standard neural net classifier training with cross-entropy loss, we do not discuss it in more detail.

Prediction. As mentioned in Section 2.1, at test time, we are given a feature vector $\mathbf{x}_* \in \mathbb{R}^D$ corresponding a single snapshot in time. To make a prediction for $\mathbf{x}_*$, we first compute the embedding of $\mathbf{x}_*$ given by $\mathbf{z}_* := f(\mathbf{x}_*)$. Then the predicted class probabilities are precisely given by $g(\mathbf{z}_*)$.

3.3 Learning the Encoder Network in the Pre-training Phase

We now explain how we modify SIMPLE-SCL to accommodate time steps and to encourage the learned embeddings to be diverse in capturing raw feature heterogeneity. The latter uses what we call a *nearest neighbor pairing mechanism*.

SIMPLE-SCL with time steps. To adapt SIMPLE-SCL to the setting with time steps, we treat every time step of every time series as its own "data point". To avoid confusion as we usually consider an entire time series to be a data point, we instead call each time step's data $(\mathbf{x}_i^{(\ell)}, y_i^{(\ell)})$ to be a *snapshot*. Thus, we take the "training data" of SIMPLE-SCL to be all the snapshots: $\bigcup_{i=1}^{N} \bigcup_{\ell=1}^{L_i} \{(\mathbf{x}_i^{(\ell)}, y_i^{(\ell)})\}$.

Nearest neighbor pairing mechanism. Whereas in our description of SIMPLE-SCL earlier (which did not have time steps) that considered two points to be similar if they share the same classification outcome, we now instead consider two *snapshots* to be similar if they share the same classification outcome *and* their raw feature vectors are "close to each other". We use a sampling approach to finding pairs of snapshots that we deem similar:

1. Initialize the set of snapshot pairs to be empty: $\mathcal{E} \leftarrow \emptyset$.
2. For each class $c \in [C]$:
   (a) Let the set $\mathcal{A}_c$ consist of all snapshots whose label is c.
   (b) While $|\mathcal{A}_c| \geq 2$:
      i. Let $(\mathbf{x}_i^{(\ell)}, y_i^{(\ell)})$ be a randomly chosen snapshot from $\mathcal{A}_c$.
      ii. (Nearest neighbor search) Among all the other snapshots in $\mathcal{A}_c$, find the one whose feature vector is closest to $\mathbf{x}_i^{(\ell)}$ (e.g., using Euclidean distance). Denote the resulting snapshot found as $(\mathbf{x}_{i'}^{(\ell')}, y_{i'}^{(\ell')})$.
      iii. Add the snapshot pair $\big((\mathbf{x}_i^{(\ell)}, y_i^{(\ell)}),\ (\mathbf{x}_{i'}^{(\ell')}, y_{i'}^{(\ell')})\big)$ to $\mathcal{E}$.
      iv. Remove $(\mathbf{x}_i^{(\ell)}, y_i^{(\ell)})$ and $(\mathbf{x}_{i'}^{(\ell')}, y_{i'}^{(\ell')})$ from $\mathcal{A}_c$.

To sample a minibatch of B data points for minibatch gradient descent, where we assume that B is even, we randomly choose $B/2$ pairs from the set $\mathcal{E}$; denote the set of these $B/2$ pairs as $\mathcal{E}_{\text{batch}}$. Note that the $B/2$ pairs in $\mathcal{E}_{\text{batch}}$ correspond to a total of B different snapshots; denote the set of these B snapshots as $\mathcal{V}_{\text{batch}}$. Then the loss we use for minibatch gradient descent during pre-training is

$$L_{\text{SCL-snapshots}} := - \sum_{((\mathbf{x}_i^{(\ell)}, y_i^{(\ell)}), (\mathbf{x}_{i'}^{(\ell')}, y_{i'}^{(\ell')})) \in \mathcal{E}_{\text{batch}}} \Gamma((i,\ell),(i',\ell');\tau),$$

where

$$\Gamma((i,\ell),(i',\ell');\tau) := \log \left[\frac{\exp(\langle f(\mathbf{x}_i^{(\ell)}), f(\mathbf{x}_{i'}^{(\ell')})\rangle/\tau)}{\sum_{\substack{(\mathbf{x}_{i''}^{(\ell'')}, y_{i''}^{(\ell'')}) \in \mathcal{V}_{\text{batch}} \\ \text{s.t. } (i'',\ell'') \neq (i,\ell)}} \exp(\langle f(\mathbf{x}_i^{(\ell)}), f(\mathbf{x}_{i''}^{(\ell'')})\rangle/\tau)} \right].$$

Ablation. Later on in our experiments, we conduct ablation experiments where we do not use nearest neighbor pairing. The only change is that in step 2(b), steps i. and ii. are replaced by randomly choosing two different snapshots $(\mathbf{x}_i^{(\ell)}, y_i^{(\ell)})$ and $(\mathbf{x}_{i'}^{(\ell')}, y_{i'}^{(\ell')})$ from $\mathcal{A}_c$ to pair up (e.g., uniformly at random).

Static outcome case. In the dynamic outcome case (i.e., the classification outcome changes over time), we use the above procedure as stated. However, in the static outcome case (i.e., the classification label is for the final time step), during the pre-training phase, we only use the snapshot corresponding to the final time step per training time series. This is because we are not sure of what the true labels should be prior to the final time step, so for now we focus learning the embeddings based on time steps where we are know the labels.

Data augmentation. When data augmentation is available, the above procedure can still be run as stated using an augmented training dataset. For example, we can keep track of what the original N raw time series are prior to any data augmentation and per epoch, we use a different random augmentation of every original training time series (and treat the augmented set of N time series as a "fresh" set of training data for that epoch). Of course, we could also do what is done in the original SCL that further pairs up two random augmentations of the same raw input (this would involve, for each epoch, generating two random augmentations per training time series). In fact, combining these two strategies for contrastive learning on images has been previously done (Dwibedi et al. 2021); however, this earlier work finds nearest neighbors in the embedding space rather than the raw feature space and has not been extended to variable-length irregularly sampled temporal data like the tabular time series data we experiment on later.

3.4 Joint Optimization of Encoder and Temporal Networks

In the second stage of training a TEMPORAL-SCL model, we jointly train the encoder and temporal networks by minimizing the overall loss

$$L_{\text{overall}} = L_{\text{SCL-snapshots}} + \alpha L_{\text{temp-reg}}, \tag{4}$$

where $L_{\text{temp-reg}}$ is a temporal smoothness loss term (to be defined shortly), and $\alpha \geq 0$ is a hyperparameter that trades off between the two losses on the right-hand side. Recall that for the i-th training data point, we previously defined the time duration $\delta_i^{(\ell)} := t_i^{(\ell+1)} - t_i^{(\ell)}$ for $\ell = 1, 2, \ldots, L_i - 1$. Moreover, h takes as input the sequence $(\mathbf{z}_i^{(1)}, \delta_i^{(1)}), \ldots, (\mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)})$ and outputs a prediction for $\mathbf{z}_i^{(\ell+1)}$. We ask that $\mathbf{z}_i^{(\ell+1)}$ and $h\big((\mathbf{z}_i^{(1)}, \delta_i^{(1)}), \ldots, (\mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)})\big)$ be close in terms of squared Euclidean distance, across all data points and time steps:

$$L_{\text{temp-reg}} := \frac{1}{N} \sum_{i=1}^{N} \frac{1}{L_i - 1} \sum_{\ell=1}^{L_i - 1} \big\| h\big((\mathbf{z}_i^{(1)}, \delta_i^{(1)}), \ldots, (\mathbf{z}_i^{(\ell)}, \delta_i^{(\ell)})\big) - \mathbf{z}_i^{(\ell+1)} \big\|^2.$$

Static outcome case. Once again, in the dynamic outcome case, we use the above procedure as stated. In the static out-

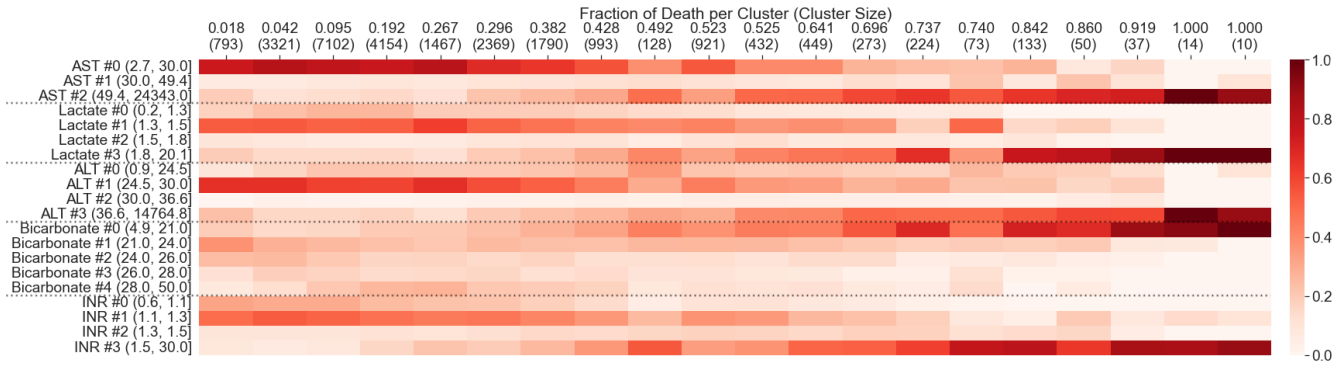


Figure 3.1: Heatmap showing how features (rows) vary across clusters (columns) for the sepsis cohort of the MIMIC dataset. Heatmap intensity values can be thought of as the conditional probability of seeing a feature value (row) conditioned on being in a cluster (column); these probabilities are estimated using test set snapshots. Columns are ordered left to right in increasing fraction of test set snapshots that come from a time series that has a final outcome of death.

come case, for this joint optimization phase, we use all snapshots (unlike during pre-training); as a reminder, snapshots that do not correspond to the final time step of a time series has the classification label "?". The idea is that now that we are accounting for temporal structure, despite us not knowing the labels prior to the final time step, we still want to encourage temporal smoothness of embeddings across time steps.

3.5 A Clustering-based Approach to Visualizing Embeddings

To find common patterns in the learned embedding space, we cluster on the different snapshots' embeddings (the $\mathbf{z}_i^{(\ell)}$ variables). We have found standard agglomerative clustering (Murtagh and Contreras 2012) to work reasonably well here and this approach trivially allows the user to adjust the granularity of clusters as needed, even from fitting the clustering model once. The idea is that we start with every snapshot's embedding as its own cluster and keep merging the closest two clusters (e.g., using complete linkage to decide on which two clusters are closest) until we are left with a single cluster that contains all the snapshots' embeddings. In conducting this procedure, we make sure to store every intermediate clustering result. Then the user could use our visualization strategy (to be described next) with any intermediate clustering result, ranging from fine- to coarse-grain clusters.

To visualize *any* clustering assignment of the snapshot embeddings (the clustering method need not be agglomerative clustering), we make a heatmap inspired by Li et al. (2020), where columns correspond to different clusters and rows correspond to different features.[1] The intensity value at the i-th row and j-th column is the fraction of test set snapshots that have the i-th row's feature value among test set snapshots in the j-th column's cluster (i.e., a conditional probability estimate of seeing a feature value given being in a cluster). An example is shown in Figure 3.1; details on the dataset used and model training are in Section 4.2. We show the "top" 5 features, where we rank features based on the maximum observed difference across clusters.[2]

From Figure 3.1, we can readily see how the features vary across clusters. For example, higher values for AST, lactate, ALT, and INR are associated with higher risk of death, whereas lower values of bicarbonate are associated with higher risk of death. There are two clusters that have fraction of death close to each other (0.737 and 0.740) but that have different patient characteristics (the biggest difference between the two clusters appears to be in lactate levels).

4 Experiments

We benchmark TEMPORAL-SCL on tabular time series data: a synthetic dataset with known ground truth embedding space structure (Section 4.1), and two standard real clinical datasets for which the embedding space is unknown (Section 4.2). No data augmentation is used. We also examine how TEMPORAL-SCL works without pre-training and, separately, without nearest neighbor pairing.

Baselines. As baselines, we use logistic regression, an LSTM (Hochreiter and Schmidhuber 1997), RETAIN (Choi et al. 2016), Dipole (Ma et al. 2017), a BERT-based transformer (Devlin et al. 2019), AC-TPC (Lee and Van Der Schaar 2020), and SIMPLE-SCL (treating snapshots as separate data points).

Experimental setup. For all datasets, we randomly split the data into 60% training, 20% validation, and 20% test sets. We train each method on the training set, tune hyperparameters based on the validation set, and report evaluation metrics on the test set. Furthermore, we choose 10 different random seeds to randomize the parameter initialization of all the models evaluated as well as randomizing the train/-validation/test sets for each experimental repeat. Details on

[1] Note that we discretize continuous features into bins. This discretization is only used during visualization and not used when training the TEMPORAL-SCL model.

[2] Per row in the heatmap, compute the difference between the largest and smallest intensity values across the row, and then rank rows using these differences, where we keep rows that correspond to the same underlying feature together.

training including hyperparameter grids are in the appendix.
Evaluation metrics. We use one-vs-rest area under the receiver operating characteristic curve (AUROC) and area under the precision-recall curve (AUPRC). For the synthetic data experiment, we also look at the fraction of experimental repeats that a method recovers the ground truth structure.

4.1 Synthetic Data

Data. We generate a 2D dataset where every time series has exactly 3 time steps. For simplicity, we only consider the static outcome case so that each time series has a single label (one of two classes: red or blue). The points are all on a 2D circle, where the only four possible time series in the embedding space are shown in Figure 4.1(a). For example, one possible time series is "● → ▼ → ★". There are a total of 10 true embedding locations (which could be thought of as cluster centers). *Note that we take the embedding space and raw feature space to be the same.* When we generate synthetic time series, each point is based on one of the 10 true ground truth embedding locations with Gaussian noise added. We randomly sample 200 of each of the 4 possible trajectories so that we have a total of 800 time series. See the appendix for details.
Experimental results. For all methods, we use a 3D embedding space (despite the ground truth one being 2D) as we found that all methods perform much worse when constrained to a 2D embedding space (we believe that this is a special case of the more general phenomenon of over-specification as documented by Livni, Shalev-Shwartz, and Shamir (2014) that training neural networks larger than needed is actually easier). Figure 4.1(b)-(d) shows the test data projected onto the 3D embedding space learned by a few methods. Note that the transformer's embedding space is not constrained to be on the unit hypersphere. We report test set accuracy scores and how often each method recovers all 10 clusters (across 10 experimental repeats; we only visually examine whether 10 clusters are clearly found and disregard the precise location of the clusters) in Table 4.1. The main finding from Table 4.1 is that while nearly all methods achieve the same (best) accuracy scores, TEMPORAL-SCL (the full version with pre-training and nearest neighbor pairing) is the only method that recovers all 10 embedding space clusters. More details, including visualizations of learned embedding spaces of all methods, are in the appendix.

4.2 Real Clinical Data

Data. We use two standard datasets:

- **MIMIC** (static outcome case)**.** We use time series data of septic patients from the MIMIC-III dataset (v1.4) (Johnson et al. 2018). We follow the same procedure as done by Komorowski et al. (2018) to identify 18,354 septic patients among which there is an observed mortality rate just above 20% (determined by death within 48h of the final observation $y_i^{(L_i)} = 1$ or death within 90 days of the final observation $y_i^{(L_i)} = 2$).
- **ADNI** (dynamic outcome case)**.** We also test our method on the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset (Petersen et al. 2010). This dataset consists of a total of 11,651 hospital visits from 1,346 patients which tracks the progression of Alzheimer's disease via follow-up observations at 6 month intervals.

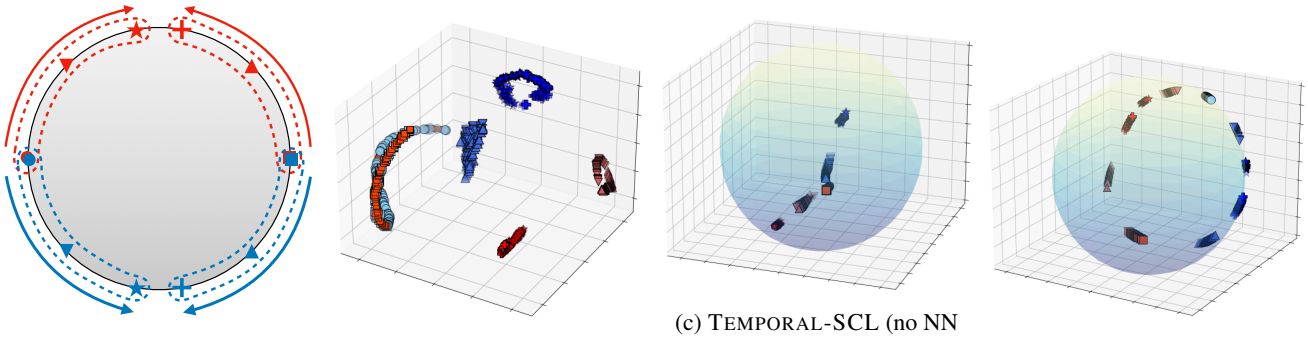
Table 4.1: Synthetic data test set accuracy (mean ± std. dev. across 10 experiments) and fraction of experiments with correct embedding structure recovery.

Model	AUROC	AUPRC	Recovery
Logistic Regression	0.902±0.010	0.900±0.003	0/10
LSTM	0.951±0.008	0.948±0.002	0/10
RETAIN	0.951±0.008	0.948±0.002	2/10
DIPOLE	0.951±0.008	0.948±0.002	3/10
AC-TPC	0.951±0.008	0.948±0.002	0/10
Transformer:BERT	0.951±0.008	0.948±0.002	1/10
SIMPLE-SCL	0.805±0.007	0.804±0.002	0/10
TEMPORAL-SCL (no pretrain)	0.950±0.009	0.944±0.004	2/10
TEMPORAL-SCL (no NN pairing)	0.951±0.008	0.948±0.002	0/10
TEMPORAL-SCL (full)	0.951±0.008	0.948±0.002	**10/10**

Experimental results. We report the prediction accuracy of the different methods evaluated in Table 4.2. For the MIMIC dataset, TEMPORAL-SCL has prediction accuracy on par with the best performing baselines. For the ADNI dataset, TEMPORAL-SCL also shows competitive results among the top-performing experimented baselines. The earlier heatmap visualization (Figure 3.1) is for the full TEMPORAL-SCL model trained on MIMIC. For the sepsis cohort in MIMIC, we observe that our clusters have a high correlation with increasing risk of death where for higher mortality risk we observe abnormal AST, Lactate, ALT, Bicarbonate, and INR values. Our findings align with the sepsis clinical literature (Nesseler et al. 2012; Villar, Short, and Lighthall 2019) where these biomarkers have correlation with increased patient mortality. In addition, we observe that for higher risk columns with similar death rates, features are more heterogeneous which highlights our model's capability in going beyond risk stratification and providing clinical insight for identifying phenotypes that have homogeneous risk. A more complete version of this heatmap (including all the features), a similar heatmap visualization for ADNI, and details on clustering are in appendix. For TEMPORAL-SCL, excluding pre-training results in a large drop in accuracy. If we instead exclude nearest neighbor pairing, we see that accuracy is largely the same for MIMIC but is noticeably worse for ADNI. In the appendix, we also show that using unsupervised metrics, the full TEMPORAL-SCL significantly outperforms the variants without pre-training and, separately, without nearest neighbor pairing.

5 Discussion

Our extension of supervised contrastive learning to handle temporal dynamics crucially uses nearest neighbor pairing to encourage embeddings to be far apart when the raw inputs are far apart, even if these raw inputs have the same classification label. Better understanding when and why nearest neighbor pairing works in different applications would be an interesting direction for future research. Meanwhile, for



(a) Ground truth (b) Transformer (c) TEMPORAL-SCL (no NN pairing) (d) TEMPORAL-SCL

Figure 4.1: Synthetic dataset: (a) the only four possible time series trajectories (each true embedding state has a unique color and shape combination), where every time series has 3 time steps, and each time series belongs to one of two classes (red or blue); (b) unsuccessful recovery of the structure by the transformer model; (c) unsuccessful recovery by TEMPORAL-SCL when nearest neighbor pairing is not used; and (d) the correct structure recovered by the full TEMPORAL-SCL.

Table 4.2: Real data test set accuracy (mean ± std. dev. across 10 experiments).

Model	MIMIC dataset		ADNI dataset	
	AUROC	AUPRC	AUROC	AUPRC
Logistic Regression	0.745±0.003	0.499±0.008	0.845±0.006	0.676±0.009
LSTM	0.767±0.003	0.509±0.005	0.947±0.002	0.823±0.005
RETAIN	0.730±0.010	0.431±0.006	0.884±0.012	0.795±0.016
DIPOLE	0.767±0.004	0.453±0.003	0.958±0.006	0.824±0.009
AC-TPC	0.703±0.006	0.432±0.007	0.839±0.013	0.681±0.017
Transformer:BERT	**0.769±0.005**	0.509±0.003	0.959±0.002	**0.922±0.003**
SIMPLE-SCL	0.744±0.003	0.486±0.003	0.902±0.024	0.796±0.020
TEMPORAL-SCL (no pretrain)	0.725±0.042	0.471±0.001	0.867±0.035	0.766±0.050
TEMPORAL-SCL (no NN pairing)	0.767±0.005	0.509±0.003	0.894±0.062	0.807±0.045
TEMPORAL-SCL (full)	0.763±0.001	**0.510±0.002**	**0.961±0.001**	0.867±0.006

simplicity, in this paper we only considered making predictions for a single time step's worth of information at a time. Future work could consider making predictions for variable-length time series. Lastly, we point out that our heatmap visualization could extend to video data (each snapshot is for a video frame) if the rows are replaced by discrete visual concepts (e.g., faces, cats, etc) that could be detected as being present or not in an image.

References

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *International Conference on Machine Learning*.

Chen, X.; Fan, H.; Girshick, R.; and He, K. 2020b. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*.

Choi, E.; Bahadori, M. T.; Sun, J.; Kulas, J.; Schuetz, A.; and Stewart, W. 2016. RETAIN: An interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems*.

Dave, I.; Gupta, R.; Rizve, M. N.; and Shah, M. 2022. TCLR: Temporal contrastive learning for video representation. *Computer Vision and Image Understanding*.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the North American Chapter of the Association for Computational Linguistics*.

Dwibedi, D.; Aytar, Y.; Tompson, J.; Sermanet, P.; and Zisserman, A. 2021. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *International Conference on Computer Vision*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural Computation*, 9(8): 1735–1780.

Johnson, A. E.; Stone, D. J.; Celi, L. A.; and Pollard, T. J. 2018. The MIMIC Code Repository: Enabling reproducibility in critical care research. *Journal of the American Medical Informatics Association*, 25(1): 32–39.

Khosla, P.; Teterwak, P.; Wang, C.; Sarna, A.; Tian, Y.; Isola, P.; Maschinot, A.; Liu, C.; and Krishnan, D. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*.

Komorowski, M.; Celi, L. A.; Badawi, O.; Gordon, A. C.; and Faisal, A. A. 2018. The artificial intelligence clinician

learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11): 1716–1720.

Lee, C.; and Van Der Schaar, M. 2020. Temporal phenotyping using deep predictive clustering of disease progression. In *International Conference on Machine Learning*.

Lee, C.; Yoon, J.; and Van Der Schaar, M. 2019. Dynamic-DeepHit: A deep learning approach for dynamic survival analysis with competing risks based on longitudinal data. *IEEE Transactions on Biomedical Engineering*, 67(1): 122–133.

Li, L.; Zuo, R.; Coston, A.; Weiss, J. C.; and Chen, G. H. 2020. Neural Topic Models with Survival Supervision: Jointly Predicting Time-to-Event Outcomes and Learning How Clinical Features Relate. In *International Conference on Artificial Intelligence in Medicine*, 371–381. Springer.

Livni, R.; Shalev-Shwartz, S.; and Shamir, O. 2014. On the computational efficiency of training neural networks. *Advances in Neural Information Processing Systems*.

Ma, F.; Chitta, R.; Zhou, J.; You, Q.; Sun, T.; and Gao, J. 2017. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Murtagh, F.; and Contreras, P. 2012. Algorithms for hierarchical clustering: an overview. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

Nesseler, N.; Launey, Y.; Aninat, C.; Morel, F.; Mallédant, Y.; and Seguin, P. 2012. Clinical review: the liver in sepsis. *Critical Care*, 16(5): 1–8.

Oord, A. v. d.; Li, Y.; and Vinyals, O. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Petersen, R.; Aisen, P.; Beckett, L.; Donohue, M.; Gamst, A.; and Harvey, D. 2010. Alzheimer's Disease Neuroimaging Initiative (ADNI). *Neurology*, 74(3): 201.

Seymour, C. W.; Kennedy, J. N.; Wang, S.; Chang, C.-C. H.; Elliott, C. F.; Xu, Z.; Berry, S.; Clermont, G.; Cooper, G.; and Gomez, H. 2019. Derivation, validation, and potential treatment implications of novel clinical phenotypes for sepsis. *JAMA*, 321(20): 2003–2017.

Villar, J.; Short, J. H.; and Lighthall, G. 2019. Lactate predicts both short-and long-term mortality in patients with and without sepsis. *Infectious Diseases: Research and Treatment*, 12: 1178633719862776.

A Appendix

A.1 Baselines

In this section we describe the baselines used for our experiments.

Logistic regression. For this simple baseline, our training data has each timestep as an individual datapoint with the true label of each timestep being the final timestep outcome of its corresponding sequence. We run logistic regression on raw features of every timestep in the training set.

Long short-term memory (LSTM). We train a multilayer LSTM on the sequences of the raw features in the training set and their corresponding outcome label and take the hidden layer of the final timestep and pass it through a linear layer to predict the outcome of the corresponding sequence. After the training is completed, we run our model on the validation set and get the probability of the positive outcome (not surviving) for each sequence.

Actor-Critic Temporal Predictive Clustering (AC-TPC). Lee and Van Der Schaar (2020) proposed an actor-critic approach from reinforcement learning for temporal predictive clustering (AC-TPC) where each cluster consists of patients with similar future outcomes of interest. In this approach an RNN-based encoder and multi-layer perceptron predictor network are first trained to initialize time series embeddings. After initialization, a selector network and embedding dictionary are jointly optimized with the encoder and predictor networks to obtain a representative embedding of each timestep for the outcome in the subsequent timestep. At inference, at each timestep, the encoder maps a sequence into a latent embedding, and the selector network assigns a cluster to this latent embedding. The centroid of the cluster assigned to the latent embedding which is stored in the embedding dictionary is then used by the predictor network to predict the future outcome of interest.

Reverse Time Attention Model (RETAIN). Choi et al. (2016) proposed an interpretable predictive model for healthcare using reverse time attention mechanism. Their proposed model is based on a two-level neural attention model that detects influential past visits and significant clinical variables within those visits by mimicking physician practice to attend to the EHR data in a reverse time order so that recent clinical measurements are likely receive higher attention.

Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks (Dipole). Ma et al. (2017) propose a model that utilizes bidirectional recurrent neural networks to remember all the information of both the past visits and the future visits. The authors introduce three attention mechanisms to measure the relationships of different visits for the prediction. Dipole uses the attention mechanism to interpret the prediction results.

Transformer: Bidirectional Encoder Representations from Transformers (BERT). We train a BERT-based (Devlin et al. 2019) model from scratch to encode the time-series data of our experiments to predict the final outcome of each time-series.

A.2 Relating SIMPLE-SCL to the Original SCL

The original version of SCL has an additional *data augmentation* step: in the loss function L_{SCL}, instead of using the batch of B points $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_B, y_B)$, we instead use a freshly generated batch of $2B$ points $(\mathbf{x}'_1, y'_1), \ldots, (\mathbf{x}'_{2B}, y'_{2B})$. Specifically, each original data point $\mathbf{x}_i$ is randomly augmented once to get $\mathbf{x}'_{2i-1}$ and then $\mathbf{x}_i$ is randomly augmented a second time to get $\mathbf{x}'_{2i}$; the augmented points $\mathbf{x}'_{2i-1}$ and $\mathbf{x}'_{2i}$ have labels y'_{2i-1} and y'_{2i} that are set to be the same as y_i.

A.3 Clustering on the Embedding Space

Hierarchical Clustering For clustering on the embedding space, we use complete linkage Agglomerative Hierarchical Clustering. We first get the embedding representation of our training dataset and then train the clustering algorithm for $K \in \{5, 7, 8, 9, 10, 14, 20\}$. Then, for predictive cluster assignment, we train a 3-Nearest Neighbor classifier (KNN) on the training data, using the cluster assignments obtained from the previous step. We use our classifier to get the predicted cluster assignment for the validation set.

A.4 Dataset Information

The following subsections provide detailed information of our synthetic experiment dataset generation as well as our real-world datasets where we used to evaluate our models on.

Synthetic Data We generate a 2D dataset where every time series has exactly 3 time steps. We consider the static outcome case so that each time series has a single label (one of two classes: red or blue). The points are all on a 2D circle, where the only four possible time series in the embedding space are shown in Figure 4.1(a) and include: (i) "● → ▼ → ★" and (ii) "■ → ▲ → +" for the blue class and iii) "● → ▼ → ★" and (iv) "■ → ▲ → +" for the red class. We place the starting time steps ("●", "■", "●", "■") on angular points $\{0°, 180°, 0°, 180°\}$ of the cicle circumference as shown on Figure 4.1(a). For these starting points we purposefully have the starting points of the two different classes fall in the same region. Next for the second time steps ("▼", "▲", "▼", "▲"), we place them on the angular points of $\{45°, 135°, -45°, -135°\}$. And finally for the terminal time steps ("★", "+", "★", "+"), we place them on angular points of $\{80°, 100°, -80° - 100°\}$. Given the clashing of the two starting time steps, there are a total of 10 true embedding locations (which could be thought of as cluster centers). When we generate synthetic time series, each point is based on one of the 10 true ground truth embedding locations with Gaussian noise ($\mathcal{N}(0, 8)$) added to the true angles noted above. We randomly sample 200 of each of the 4 possible 3-time stepped trajectories so that we have a total of 800 time series.

Medical Information Mart for Intensive Care (MIMIC) We consider the trajectory of septic patients using data from the Medical Information Mart for Intensive Care (MIMIC-III) dataset (v1.4) (Johnson et al. 2018). We follow the same procedure as done by Komorowski et al. (2018) to identify

18,354 septic patients among which there is an observed mortality rate just above 20% (determined by death within 48h of the final observation $y_i^{(T_i)} = 1$ or death within 90 days of the final observation $y_i^{(T_i)} = 2$). From MIMIC, we extract demographic and physiological features according to Seymour et al. (2019), resulting in 29 features. Note that Seymour et al. group these features as follows: (1) Hepatic: Bilirubin, AST, ALT; (2) Hematologic: Hemoglobin, INR, Platelets; (3) Neurologic: GCS; (4) Cardiovascular: Heart rate, Systolic blood pressure, Bicarbonate, Troponin, Lactate; (5) Pulmonary: Respiratory rate, SaO2, PaO2; (6) Inflammatory: Temperature, ESR, WBC count, Bands, C-Reactive protein; (7) Renal: Serum creatinine; (8) Other: Age, Gender, Elixhauser, Albumin, Chloride, Sodium, Glucose, BUN. We compile the measurements recorded of these features in MIMIC for every 4 hours (duration of each timestep). We impute the missing values from the population median if a measurement is not recorded before and the previously recorded value if the measurement is recorded in a previous timestep. We also include a set of 26 indicators for our time-varying features at each timestep that tracks whether a measurement was recorded ($= 1$) or imputed ($= 0$). This dataset represent the case where only the final outcome-label is known and non-terminal timesteps have unknown state-label.

Alzheimer's Disease Neuroimaging Initiative (ADNI) We also test our method on the Alzheimer's Disease Neuroimaging Initiative (ADNI) dataset (Petersen et al. 2010). This dataset consists of a total of 11,651 hospital visits from 1,346 patients which tracks the progression of Alzheimer's disease via follow-up observations at 6 months interval. Each patient has 21 variables out of which 5 are static and 16 are time-varying. The features include information on demographics, biomarkers of the brain function, and cognitive test results. Following Lee and Van Der Schaar (2020), we set our predictions on three diagnostic groups of normal brain functioning ($y_i^{(T_i)} = 0$), mild cognitive impairment ($y_i^{(T_i)} = 1$), and Alzheimer's disease ($y_i^{(T_i)} = 2$) which is known at every timestep. This dataset represent the case where we know the outcome-label at both terminal and non-terminal timesteps.

A.5 Missing Data Imputation

For both real datasets of this paper, we take the following approach to impute the missing features in our experiments. (1) For each patient, if at any time step feature has been recorded in a previous time step, we use the last recorded value to replace the missing feature. (2) If at no previous time step a feature is recorded, we calculate the population median of the feature among all patients and impute the missing feature with this value. In the future work, we aim to extend our approach to be similar to that of Seymour et al. (2019), where multiple imputation with chained equations was used to account for missing data.

A.6 Training Hyperparameters

Table A.1 summarizes the hyperparameters used in all our experimentations presented in this paper.

A.7 Performance Metrics

We use the same evaluation metrics as Lee and Van Der Schaar (2020). To evaluate the supervised performance of our model and the baselines, we use area under receiver operator characteristic curve (AUROC) and area under precision-recall curve (AUPRC) obtained from the label predictions of our model and the ground-truth labels on the outcomes of interest.

In the ablation studies presented in Section A.10, we also evaluate the unsupervised performance of different ablations of TEMPORAL-SCL. We cluster on the learned hyperspherical embeddings using a complete linkage Agglomerative Hierarchical Clustering to discover discrete latent states that could be of interest. We utilize three standard metrics for the scenario when ground-truth label is known and a fourth completely unsupervised metric. The metrics used are purity score, normalized mutual information (NMI), adjusted rand index (ARI), and silhouette index (SI). Purity score ranges from 0 to 1 and explains the homogeneity of each cluster with regards to the labels. NMI ranges from 0 to 1 and is an information theoretic measure of mutual information shared between the labels and the cluster and is adjusted for the number of clusters with 1 being perfect clustering. ARI ranges from -1 to 1 and evaluates the percentage of correct cluster assignments where 0 corresponds to completely random assignment and 1 being perfect clustering. Lastly, SI is a completely unsupervised metric ranging from -1 to 1 which provides a simultaneous measure of (1) how similar members of a cluster are to their own cluster capturing the homogeneity within a cluster and (2) how different members of each cluster are compared to other clusters capturing heterogeneity across different clusters.

A.8 Synthetic Data Results

The numerical quantitative results for the synthetic dataset is presented below. In section 4.1 a quantitative explanation of how our proposed model compares to the tested baselines is presented. In Figure A.1, we visualize the embedding space of all the models tested. For all the baselines tested where the embedding space does not naturally lie on the hypersphere, we show a 3D TSNE manifold of the embedding space where we qualitatively pick the best structure for each method by varying the perplexity taken from the following set: $\{5, 20, 30, 40, 50\}$. For the TEMPORAL-SCL model and its ablations where the embedding is constrained to lie on the hypersphere, we directly plot the embeddings.

As it can be seen here, our proposed model is the only model that can fully recover the structure and the 10 clusters of time steps from the input raw feature space shown in Figure 4.1(a).

A.9 Running Time Comparison

Given that there are multiple components to optimise simultaneously, we perform a comparison of how computationally intensive the training of our proposed model is

Table A.1: Training Details and Hyperparamters for Experiments.

Experiment	**Synthetic Dataset**
Main Network	2-Layer FCN ReLU Activation
Main Network Layer Size	(input-dimension)2→16→3(embedding-space)
Prediction Network	Linear Layer with Softmax Activation 3→2
Temporal Module	1-Layer LSTM 3→3
Optimization	Optimizer: Adam, Learning Rate: 1e-4, Number of Epochs: 20, Batch-Size: 32
Experiment	**MIMIC**
Main Network	2-Layer FCN ReLU Activation
Main Network Layer Size	(input-dimension)55→32→32(embedding-space)
Prediction Network	Linear Layer with Softmax Activation 32→3
Temporal Module	1-Layer LSTM 32→32
Optimization	Optimizer: Adam, Learning Rate: 1e-4, Number of Epochs: 100, Batch-Size: 128
Experiment	**ADNI**
Main Network	2-Layer FCN ReLU Activation
Main Network Layer Size	(input-dimension)21→50→16(embedding-space)
Prediction Network	Linear Layer with Softmax Activation 16→3
Temporal Module	1-Layer LSTM 16→16
Optimization	Optimizer: Adam, Learning Rate: 1e-4, Number of Epochs: 100, Batch-Size: 128

compared to one of our baselines (AC-TPC (Lee and Van Der Schaar 2020)). We report the running time of training our model compared to AC-TPC in Table A.2 for training on our machines with identical software/hardware configurations (AMD Ryzen 9 5900X with NVIDIA GTX 1080).

A.10 Ablation Study

In this section, we present an ablation study of our model to see how each modification contributes to the performance. Our full model includes pre-training with SIMPLE-SCL, enhancing the Temporal Network h, and using "labels+feature-similarity" for finding the nearest neighbor pairs. We train three additional models in this section on both our MIMIC and ADNI datasets. (1) TEMPORAL-SCL trained without the Temporal Network h which simplifies to SIMPLE-SCL, and (1) TEMPORAL-SCL trained without pre-training, (3) TEMPORAL-SCL trained with nearest neighbor pairs that use "labels only" instead of "labels+feature-similarity". Note that to discover discrete latent states that could be of interest, we cluster on the learned hyperspherical embeddings (the $\mathbf{z}_i^{(\ell)}$ variables) using a complete linkage Agglomerative Hierarchical Clustering (Murtagh and Contreras 2012). We describe our clustering approach in Appendix A.3. The unsupervised metrics presented here are evaluated for the same number of clusters in each ablation.

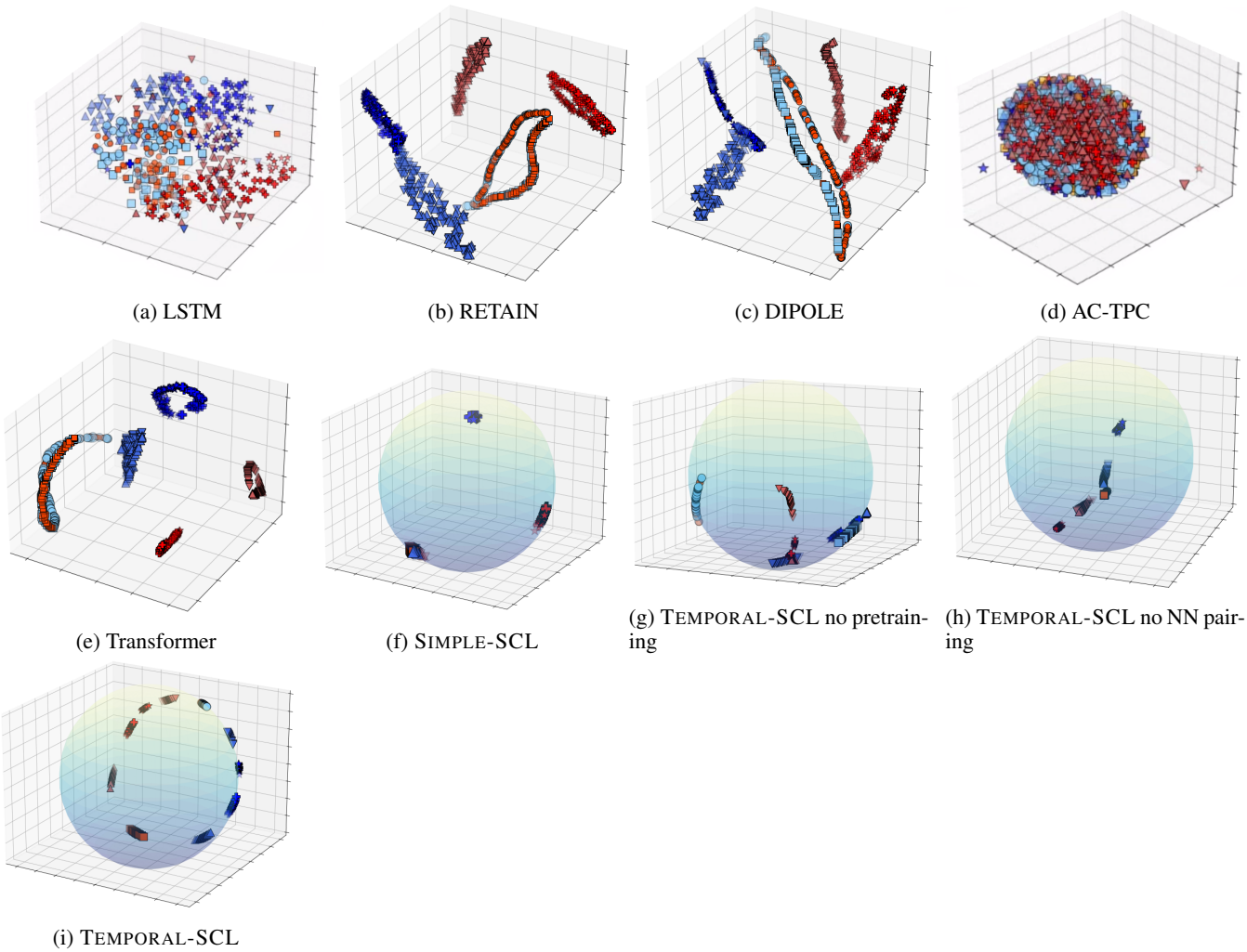
Firstly, our empirical findings show that for the models without pre-training and without the temporal network h (SIMPLE-SCL), we see a clear performance drop for all supervised and unsupervised metrics which highlights the importance of inclusion of these modules in our model training. We also calculated the Silhouette Index (SI) of "labels only" and "labels+feature similarity" in the last column of Table A.3. As it can be seen from the ablation results, with respect to the supervised prediction performance (AUROC, AUPRC), the two models perform similarly. However, the main gain of using "labels+feature similarity" comes in the unsupervised prediction performance (Purity, NMI, ARI, SI). This is especially apparent in the Silhouette Index score (a measure of how similar an object is to its own cluster compared to other clusters) where we see the greatest boost in performance when using "labels+feature similarity" instead of "labels only" where it shows how our model moves away from just stratifying risk (which is what the supervised metrics are measuring) to additionally being capable of identifying homogeneous disease phenotypes. These experiments together, underscore the significance of having the different building blocks of our model for achieving the highest performance gain in our experiments.

A.11 Visualizing Embedding Clusters for MIMIC

We show the full heatmap of Figure 3.1 for how the features vary across clusters in the test set of our MIMIC dataset in Figure A.2.

A.12 Visualizing Embedding Clusters for ADNI

To interpret each cluster for ADNI, we plot the heatmap how features (rows) vary across clusters for the test patients of ADNI in Figure A.3. Columns are ordered (left to right) in dementia rate. Here we can also see that abnormal feature values that are correlated with higher risk of dementia such as irregular dementia rating scores are present in clusters containing higher proportion of AD.



(a) LSTM (b) RETAIN (c) DIPOLE (d) AC-TPC

(e) Transformer (f) SIMPLE-SCL (g) TEMPORAL-SCL no pretraining (h) TEMPORAL-SCL no NN pairing

(i) TEMPORAL-SCL

Figure A.1: Embedding space representation of the test trajectories of simulated dataset. For (b)- (d) 3-Component TSNE Plot of Embedding Space and for (a) Hyperspherical Embedding is shown.

Table A.2: Training Run-Time

Model	**Synthetic Dataset**	**MIMIC**
AC-TPC	75 seconds	42 minutes
TEMPORAL-SCL	46 seconds	47 minutes

Table A.3: Ablation Study: Supervised and unsupervised performance.

Dataset	Model	Supervised		Unsupervised			
		AUROC	AUPRC	Purity	NMI	ARI	SI
MIMIC	SIMPLE-SCL	0.744±0.003	0.486±0.003	0.773±0.005	0.007±0.003	0.003±0.002	0.011 ± 0.010
	TEMPORAL-SCL (no pretrain)	0.725±0.042	0.471±0.001	0.775±0.000	0.007±0.002	0.002±0.001	0.031 ± 0.018
	TEMPORAL-SCL (no NN pairing)	**0.767±0.005**	0.509±0.003	0.781±0.000	0.112±0.007	**0.150±0.034**	0.143 ± 0.007
	TEMPORAL-SCL (full)	0.763±0.001	**0.510±0.002**	**0.793±0.001**	**0.115±0.002**	0.128±0.014	**0.423 ± 0.065**
ADNI	SIMPLE-SCL	0.902±0.024	0.796±0.020	0.639±0.023	0.159±0.058	0.230±0.012	0.177 ± 0.144
	TEMPORAL-SCL (no pretrain)	0.867±0.035	0.766±0.050	0.713±0.095	0.275±0.120	0.209±0.104	0.149 ± 0.080
	TEMPORAL-SCL (no NN pairing)	0.894±0.062	0.807±0.045	0.749±0.009	0.446±0.009	0.334±0.015	0.163 ± 0.008
	TEMPORAL-SCL (Full)	**0.961±0.001**	**0.867±0.006**	**0.755±0.023**	**0.452±0.031**	**0.399±0.010**	**0.259 ± 0.031**

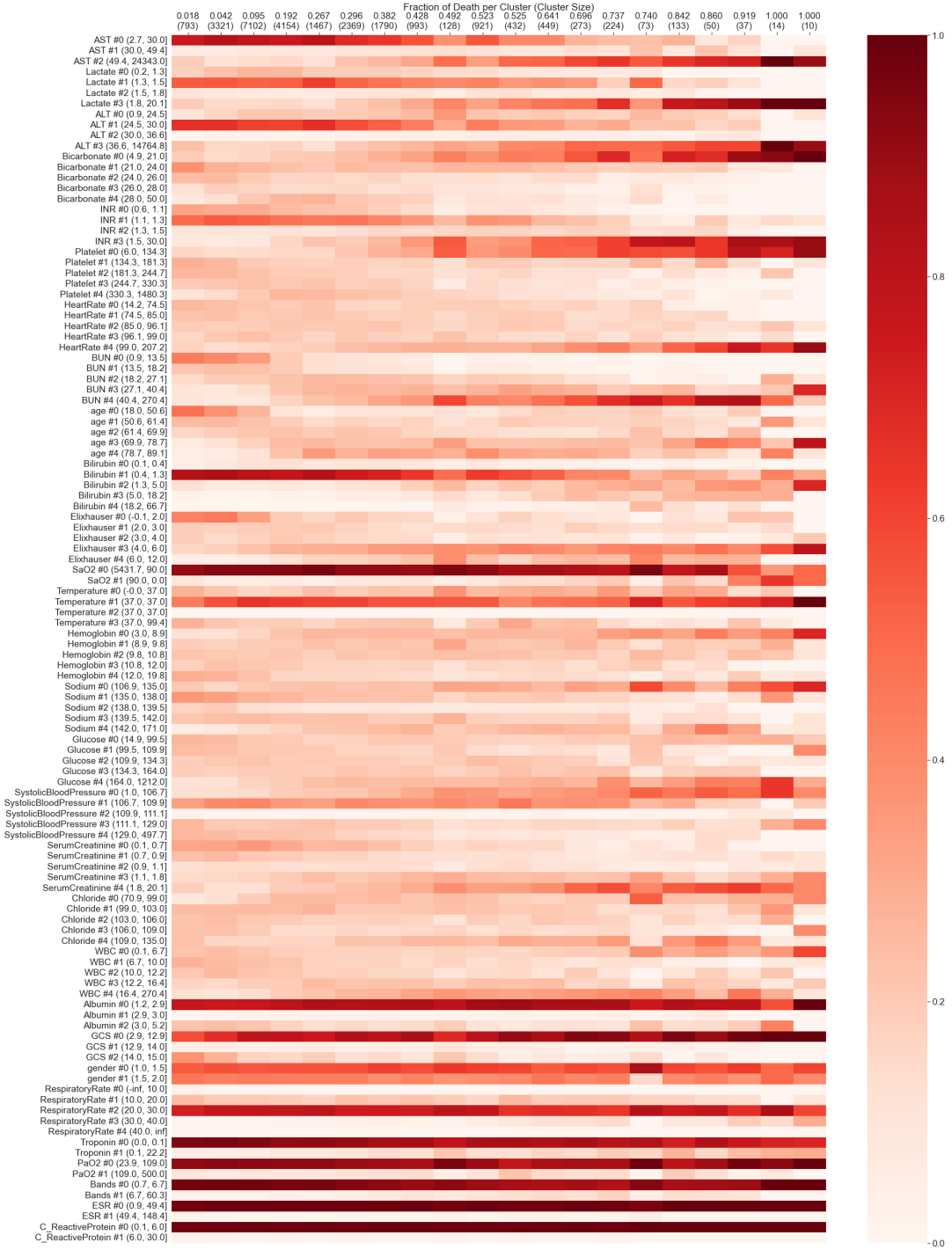


Figure A.2: Heatmap showing how features (rows) vary across clusters (columns) for the sepsis cohort of the MIMIC dataset. Heatmap intensity values can be thought of as the conditional probability of seeing a feature value (row) conditioned on being in a cluster (column); these probabilities are estimated using test set snapshots. Columns are ordered left to right in increasing fraction of test set snapshots that come from a time series that has a final outcome of death.

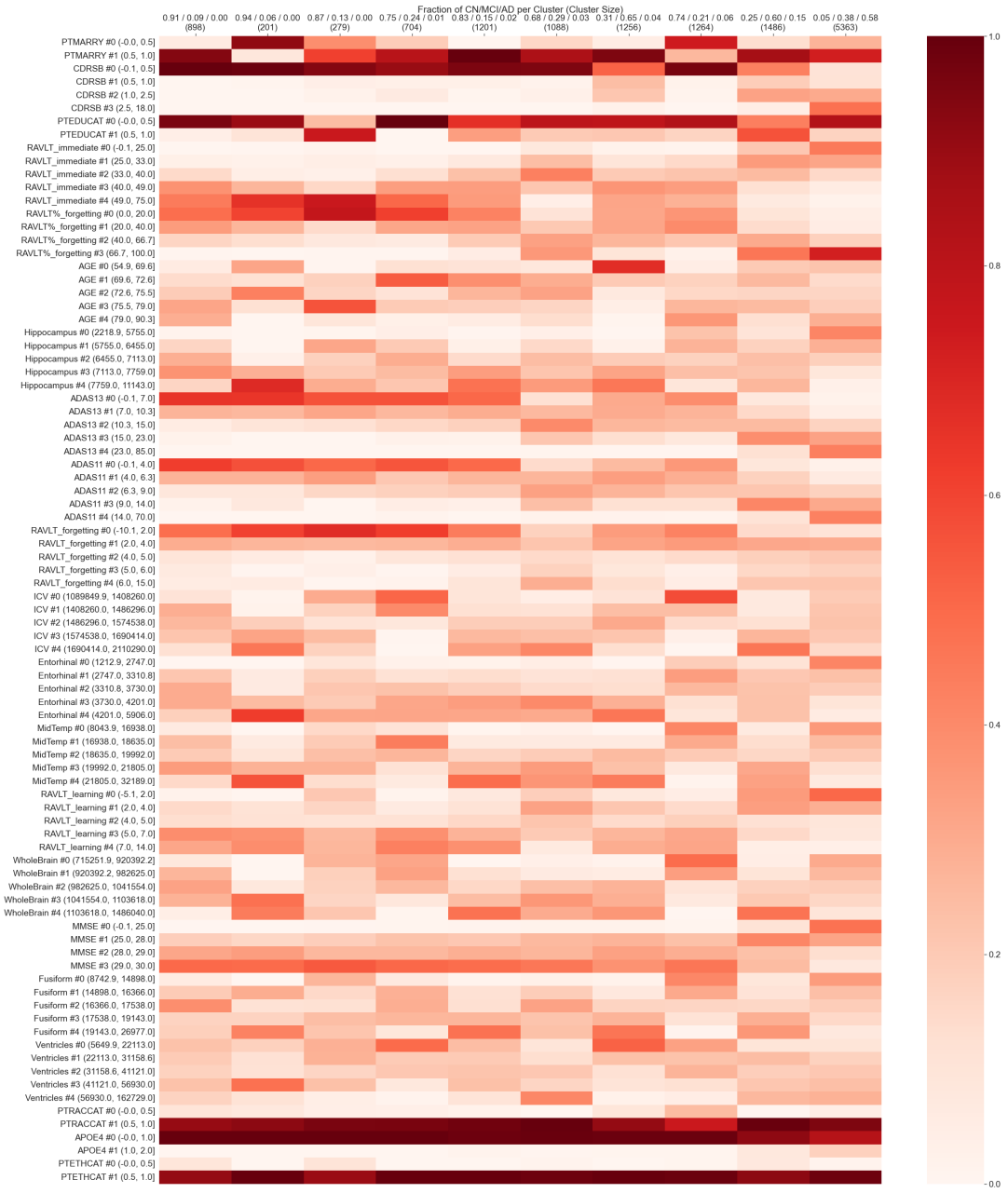


Figure A.3: Heatmap showing how features (rows) vary across clusters (columns) for the ADNI dataset. Heatmap intensity values can be thought of as the conditional probability of seeing a feature value (row) conditioned on being in a cluster (column); these probabilities are estimated using test set snapshots. Columns are ordered left to right in increasing fraction of test set snapshots that come from a time series that has a final outcome of Alzheimer's Disease.