Neighborhood Contrastive Learning Applied to Online Patient Monitoring

Hugo Yèche [* 1] Gideon Dresdner [* 1] Francesco Locatello [2] Matthias Hüser [1] Gunnar Rätsch [1]

Abstract

Intensive care units (ICU) are increasingly looking towards machine learning for methods to provide online monitoring of critically ill patients. In machine learning, online monitoring is often formulated as a supervised learning problem. Recently, contrastive learning approaches have demonstrated promising improvements over competitive supervised benchmarks. These methods rely on well-understood data augmentation techniques developed for image data which do not apply to online monitoring. In this work, we overcome this limitation by supplementing time-series data augmentation techniques with a novel contrastive learning objective which we call neighborhood contrastive learning (NCL). Our objective explicitly groups together contiguous time segments from each patient while maintaining state-specific information. Our experiments demonstrate a marked improvement over existing work applying contrastive methods to medical time-series.

1. Introduction

Recent advances in contrastive learning have shown that unsupervised learning can improve upon competitive computer vision benchmarks (Misra & Maaten, 2020; Chen et al., 2020a; He et al., 2020; Tian et al., 2020; Caron et al., 2020). Such methods rely on data augmentations to construct semantic-preserving views of samples. By aggregating them using a Noise Contrastive Estimation objective (Gutmann & Hyvärinen, 2010), contrastive learning aims to learn view-invariant representations. In subsequent work, Khosla et al. (2020) showed that a supervised extension of this objective would further extend its benefits over end-to-end training. Building on these successes, researchers ap-

plied this methodology to medical time-series data (Cheng et al., 2020; Kiyasseh et al., 2020; Mohsenvand et al., 2020).

Supervised learning approaches have enjoyed successes in online monitoring of organ failure and other life-threatening events (Hyland et al., 2020; Tomašev et al., 2019; Schwab et al., 2020; Horn et al., 2020; Li et al., 2020). On the other hand, unsupervised representation learning has not been widely applied in this setting. To our knowledge, Lyu et al. (2018) is the only work that makes an attempt. We conjecture that this is due to the additional challenges of working in this setting: difficult-to-interpret datatypes and heterogeneous distributions of samples.

Time-series are often less humanly understandable than images. Finding semantically preserving augmentations — crucial to recent advances in contrastive learning — is challenging (Um et al., 2017; Fawaz et al., 2018). In addition, biosignal data suffers from a particular domain heterogeneity problem due to multiple samples originating from the same patient. While samples from a single patient will have many commonalities, they also will exhibit changes which become important when trying to perform learning-based prediction (Morioka et al., 2015; Farshchian et al., 2018; Özdenizci et al., 2020). In the online monitoring setting, these shifts are amplified by large overlaps in history between time segments from a single patient stay.

Preliminary works (Cheng et al., 2020; Kiyasseh et al., 2020) propose sampling approaches to overcome this between-patient heterogeneity in a contrastive learning setup. However, their methods only cover the edge cases of complete dependence or non-dependence between labels and source subjects, while ignoring the factor of time. Such assumptions show limitations in online monitoring where a patient state evolves continuously. In this work, we propose a contrastive learning framework addressing the complexity of online monitoring tasks[1]. Our contributions can be summarized as follows:

1. We propose NCL, a new contrastive learning objective that can explicitly induce a prior structure to representations in a modular, user-defined manner, allowing to address the heterogeneity problem in online monitoring data.

2. We show that, when used unsupervised, our approach

[*]Equal contribution [1]Department of Computer Science, ETH Zürich, Switzerland [2]Amazon (most work was done when Francesco was at ETH Zurich and MPI-IS). Correspondence to: Hugo Yèche <hyeche@ethz.ch>, Gideon Dresdner <dgideon@ethz.ch>.

[1]https://github.com/ratschlab/ncl

shows competitive results for several online monitoring benchmarks on open-source medical datasets. In addition, it outperforms supervised learning in the limited-labeled-data setting.

3. Moreover, when supervised, we demonstrate that it outperforms end-to-end counterparts and previous supervised extensions to contrastive learning. We also show significant improvement over them in a transfer learning setting.

2. Related Work

Contrastive learning for biological signals. There is a recent burst of work applying unsupervised contrastive learning to biosignal data such as electroencephalogram (EEG) and electrocardiogram (ECG) (Banville et al., 2020; Kiyasseh et al., 2020; Cheng et al., 2020; Mohsenvand et al., 2020). The problems posed by such biosignal data have important commonalities and differences with EHR data. Unlike EHR data, biosignal data has a much higher time resolution and known spatial dependencies between channels.

Like EHR data, biosignal data exhibits between-patient heterogeneity. We show in Section 4 that for online monitoring data this problem is further aggravated as the within-patient variance of time points is smaller than the between-patient variance. Cheng et al. (2020) tackle this problem by limiting their negative sampling procedure to samples only within the same patient, whereas Kiyasseh et al. (2020) do the opposite — by exclusively sampling negatives across patients. Both Banville et al. (2020) and Franceschi et al. (2019) approach this problem by enforcing temporal smoothness between contiguous samples, similar in spirit to Mikolov et al. (2013).

Mohsenvand et al. (2020) do not pay special attention to between-patient heterogeneity. Instead, they achieve competitive results by applying recently improved neural network architectures and augmentation techniques from Chen et al. (2020a). Mohsenvand et al. (2020) provide the impetus for our work. We expand the methodology of Chen et al. (2020a) and He et al. (2020) to incorporate a contrastive learning objective capable of dealing with the patient-induced heterogeneity problem.

Unsupervised patient state representation for EHR Data. There is a growing body of work on unsupervised representation learning that learns patient-wise representations (Miotto et al., 2016; Darabi et al., 2019; Landi et al., 2020). While EHR data is often in the form of a time-series, this body of work focuses on learning a single, time-invariant representation per patient, which cannot support online monitoring tasks.

In contrast, there is little prior work on time-dependent

patient representations. To the best of our knowledge, the Seq2Seq autoencoder-based work of Lyu et al. (2018) is the only one. Their work aims to further develop patient state representation with the goal of improving fine-tuning performance in the limited labeled data setting. Other works also explore patient state-representation learning but with specific focuses such as reinforcement learning (Killian et al., 2020) or multi-task learning (McDermott et al., 2020).

3. Neighborhood Contrastive Learning (NCL)

3.1. Preliminaries

Data definition. We are given a set of patients where each patient may have multiple ICU stays. Taking the union across all patients gives a total of S patient stays. Each patient stay itself is composed of a vector of static demographic features $\mathbf{d}^p$, known at admission time, and a multivariate time-series $\mathbf{s}^p$ describing the stay.

Online monitoring aims to make a prediction given the history up to time t. Because the length-of-stay can vary considerably across patients, we define a maximum history window t_h and slide that window across the patient's time series. Each window is denoted as $\mathbf{s}_t^p = [s_{t-t_h}^p, \ldots, s_t^p]$. Our goal is to make a prediction for each $\boldsymbol{x}_t^p \triangleq (\mathbf{d}^p, \mathbf{s}_t^p)$.

Each patient stay has total history length of t^p. Taking the union of all windowed time-segments $\mathcal{S}^p = \{ (\mathbf{d}^p, \mathbf{s}_t^p) \mid t \leq t^p \}$ gives the final dataset definition $\mathcal{D} = \bigcup_{p=0}^{S} \mathcal{S}^p$.

Pipeline specification. Let $B = \{ \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \}$ denote a minibatch of N examples. To each example we associate two views $\tilde{\boldsymbol{x}}_i$ and $\tilde{\boldsymbol{x}}_{v(i)}$ which are constructed using data augmentation. This gives a total of $2N$ views in the minibatch $V = \{ \tilde{\boldsymbol{x}}_1, \ldots, \tilde{\boldsymbol{x}}_{2N} \}$.

To each view, we apply an encoder and a momentum encoder, denoted f_e and f_m respectively, to obtain representations $Z^e = \{ \boldsymbol{z}_1^e, \ldots, \boldsymbol{z}_{2N}^e \}$ and $Z^m = \{ \boldsymbol{z}_1^m, \ldots, \boldsymbol{z}_{2N}^m \}$. All representations are normalized by dividing by their Euclidean norms (projecting onto the unit sphere) (He et al., 2020; Tian et al., 2020; Wang & Isola, 2020).

During training, these representations are further projected using two distinct projectors, $h_e(\cdot)$ for Z^e and $h_m(\cdot)$ for Z^m resulting in two sets of projected representations, $P = \{ \boldsymbol{p}_1, \ldots \boldsymbol{p}_{2N} \}$ and $P^m = \{ \boldsymbol{p}_1^m, \ldots, \boldsymbol{p}_{2N}^m \}$. These projections are also normalized by their Euclidean norms.

P^m is used to update a queue of negative samples $Q = \{ \boldsymbol{q}_1, \ldots, \boldsymbol{q}_M \}$. Specifically, at each training step, the oldest $2N$ elements of Q are replaced by P^m in sliding manner. Particularly, for $k < 2N$, we have $q_k = p_k^m$. This allows for a large number of negative samples since we can choose $M \gg 2N$.

3.2. Regular Contrastive Loss

Consider the contrastive objective from Chen et al. (2020c).

$$\mathcal{L}^{\mathrm{CL}} = -\sum_{i=1}^{2N} \log \frac{\exp\left(\boldsymbol{p}_i \cdot \boldsymbol{p}_{v(i)}^m / \tau\right)}{\sum_{k \neq i}^{M} \exp\left(\boldsymbol{p}_i \cdot \boldsymbol{q}_k / \tau\right)} \quad (1)$$

where $\tau > 0$ is the temperature scaling parameter.

This objective does not take labels into account. As a result, it is possible that a sample $\boldsymbol{x}_k$ with the same label as $\boldsymbol{x}_i$ is selected as a negative sample which repels their corresponding representations. Khosla et al. (2020) proposes a supervised objective to remedy this issue. They define a loss which gives an attractive force to representations generated from examples with the same label. Their so-called Supervised Contrastive Loss (SCL) is defined as

$$\mathcal{L}^{\mathrm{SCL}} = \sum_{i=1}^{2N} \frac{-1}{|Y(i)|} \sum_{l \in Y(i)} \log \frac{\exp\left(\boldsymbol{p}_i \cdot \boldsymbol{p}_l^m / \tau\right)}{\sum_{k \neq i}^{M} \exp\left(\boldsymbol{p}_i \cdot \boldsymbol{q}_k / \tau\right)} \quad (2)$$

where $Y(i) = \{j \mid y_j = y_i\}$ indexes samples with the same label as $\boldsymbol{x}_i$.

There are two important observations to make. First, $Y(i)$ is defined arbitrarily in terms of labels, therefore the learned representation is specific to a given downstream task.

Second, SCL tends towards a degenerate solution where representations of examples with the same label collapse to become the same vector. This might lead to poor generalization to other tasks, thus reducing the performance of SCL in transfer learning. In the following section we go through the steps to generalize this term and derive the corresponding training objective.

3.3. Neighborhood-aware Loss

We say that two samples share the same neighborhood if they share some predefined attributes. For example, two words pronounced by the same speaker in speech recognition or two states of a patient that are within w hours of each other in an EHR.

We represent this as a collection of binary functions of the form $n(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which returns one when $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ are in the same neighborhood and zero otherwise. These functions provide a modular way of defining neighborhoods. The i-th neighborhood $N(i) = \{k \neq i \mid n(\boldsymbol{x}_i, \boldsymbol{x}_k) = 1\}$ is simply the set of samples for which $n(x_i, \cdot)$ returns one. Note that $n(\boldsymbol{x}_i, \boldsymbol{x}_j) = \mathbb{1}[y_i = y_j]$ gives precisely $N(i) = Y(i)$ as defined in Khosla et al. (2020).

This naturally leads to our generalization of $\mathcal{L}^{\mathrm{SCL}}$ which we call the Neighbors Alignment (NA) objective. This

objective encourages representations coming from the same neighborhood to align.

$$\mathcal{L}^{\mathrm{NA}} = \sum_{i=1}^{2N} \frac{-1}{|N(i)|} \sum_{l \in N(i)} \log \frac{\exp\left(\boldsymbol{p}_i \cdot \boldsymbol{p}_l^m / \tau\right)}{\sum_{k \neq i}^{M} \exp\left(\boldsymbol{p}_i \cdot \boldsymbol{q}_k / \tau\right)} \quad (3)$$

By itself, $\mathcal{L}^{\mathrm{NA}}$ suffers from the same drawbacks as $\mathcal{L}^{\mathrm{SCL}}$. Specifically, it will eventually result in a trivial solution in which neighborhoods collapse to a single point. To remedy this problem, we propose the Neighbor Discriminative (ND) objective:

$$\mathcal{L}^{\mathrm{ND}} = -\sum_{i=1}^{2N} \log \frac{\exp\left(\boldsymbol{p}_i \cdot \boldsymbol{p}_{v(i)}^m / \tau\right)}{\sum_{k \in N(i)} \exp\left(\boldsymbol{p}_i \cdot \boldsymbol{q}_k / \tau\right)} \quad (4)$$

This objective more closely resembles the original contrastive learning objective described in Chen et al. (2020a). It encourages representations from views of the same anchor to be similar to one another. Though, negatives used for normalization are neighbors to the anchors. This objective allows preserving a needed diversity within each neighborhood, as neighbors do not necessarily share the same downstream task label.

Our final objective is a weighted average between these two objectives called *Neighborhood Contrastive Learning*. We introduce an explicit trade-off parameter $\alpha \in [0, 1]$ to smoothly interpolate between intra- and inter-neighborhood properties of the representations:

$$\mathcal{L}^{\mathrm{NCL}} = \alpha \mathcal{L}^{\mathrm{NA}} + (1 - \alpha) \mathcal{L}^{\mathrm{ND}} \quad (5)$$
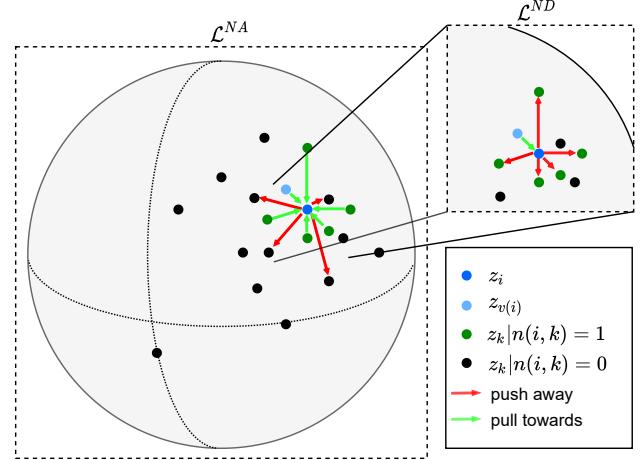

Figure 1. Illustration of our objective $\mathcal{L}^{\mathrm{NCL}}$. On the left, $\mathcal{L}^{\mathrm{NA}}$ induces embeddings from the same neighborhood (green) as the anchor $\boldsymbol{z}_i$ (dark blue) to be closer than samples external to it (black). In contrast, on the right, $\mathcal{L}^{\mathrm{ND}}$ preserves a hierarchy between neighbors (green) and the other view of the anchor's original example $z_{v(i)}$ (light blue).
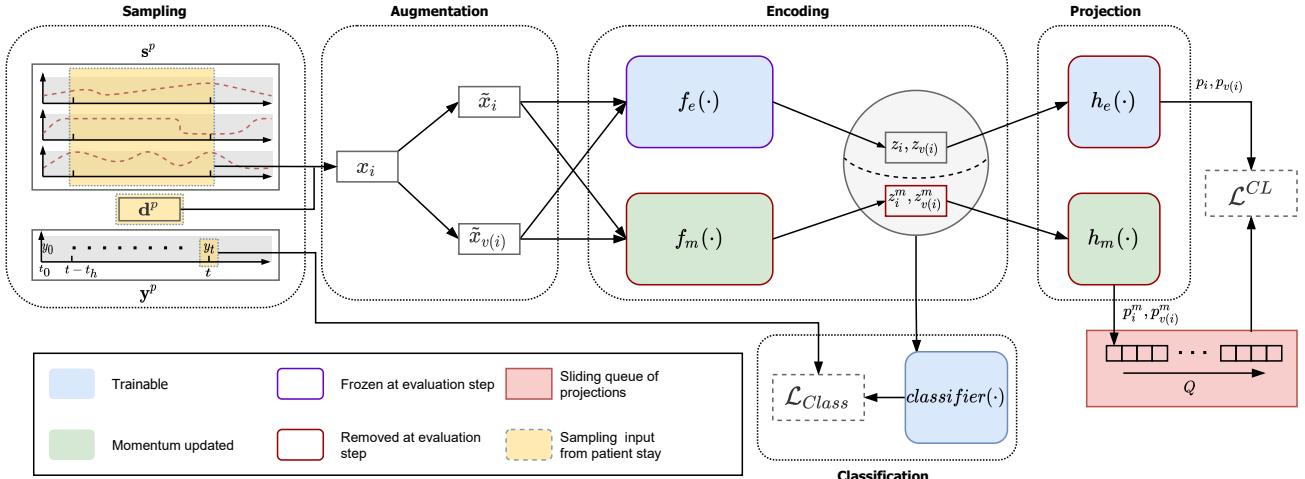
Figure 2. Schema of the contrastive pipeline initially proposed by (Chen et al., 2020c). From a patient stay p, we sample $x_i = (\mathbf{s}_t^p, \mathbf{d}^p)$ corresponding to the patient state at time t. We augment it twice and pass both views, $\tilde{x}_i$ and $x_{\tilde{v}(i)}$, through an encoder f_e and a momentum encoder f_m. At training time, the representations are further projected with h_e and h_m. From these projections and the sliding momentum queue Q, we compute the contrastive objective $\mathcal{L}^{\mathrm{CL}}$. At evaluation time, we freeze f_e and train a classifier on top of the learned representation.

4. Application to Online Monitoring

4.1. Motivations

Introduced as in Section 3, our framework, as a combination of a neighborhood function and an objective, is quite general. While many applications exist, we believe online monitoring data shows two specific attributes making it the better candidate to motivate its use.

Using the definition from Section 3.1, we note that each dataset $\mathcal{D}$ is composed of subsets of samples $\mathcal{S}^p$, all originating from a single patient stay. Moreover, elements from these subsets not only share their origin but are all time-dependent on one another. Therefore, there exists a prior structure to the data unknowingly correlated to its labeling. When using a patient-dependant neighborhood, $\mathcal{L}^{\mathrm{NA}}$ aims to preserve the patient-specific features common to all neighbors, while $\mathcal{L}^{\mathrm{ND}}$ aims to find discriminative features across neighbors.

The previous observations motivates the use of our method. However, other data types collected from human subjects, including biosignals, exhibit the same attributes. Online monitoring distinguishes itself by the existence of overlaps between examples from contiguous states of a patient. These shared history segments not only increase prior dependencies among samples, but they also limit the possible use of temporal data augmentations to mitigate the problem. Indeed, any augmentation creating identical views from two separate examples does not preserve their singularity.

To summarize, as other data originating from human sub-jects, online monitoring data shows strong distribution shifts across patients unknowingly correlated to labeling. Yet, relying on augmentation strategies to tackle the problem is limited by their need to be semantic-preserving. That motivates focusing on other components to remedy this issue as proposed in our work.

4.2. Design of neighborhood functions for online monitoring

Based on our previous observations of the online monitoring of patients state, we now detail our two proposed neighborhood functions to cope with distribution heterogeneity: (1) time-preserving (2) label-preserving neighborhood.

The first preserves the time dependency of the representations of the time-series segments. We chose to consider as neighbors samples from a patient that are close in time motivated by (Banville et al., 2020) and (Franceschi et al., 2019) works. We define a neighborhood function $n_w(\boldsymbol{x}_i, \boldsymbol{x}_j)$ with window size w. To samples $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, we can associate segments from patient stays, $\mathbf{s}_{t_i}^{p_i}$ and $\mathbf{s}_{t_j}^{p_j}$. Then $n_w(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is an indicator function which takes the value one when $p_i = p_j$ *and* $|t_i - t_j| < w$ and zero otherwise. The associated neighborhood $N_w(i)$ is defined as in Section 3. We refer to this approach as $\mathrm{NCL}(n_w)$.

The label-preserving neighborhood function is $n_Y(\boldsymbol{x}_i, \boldsymbol{x}_j)$ which takes the value of one when the two samples have the same downstream label and zero otherwise. This resembles Khosla et al. (2020) but, when inserted into our proposed contrastive learning objective (Eq. 5) still provides a balance

between label-specific and general features. We refer to the resulting learning objective as NCL(n_Y).

4.3. Contrastive Framework

We adopt the pipeline used by Cheng et al. (2020); Chen et al. (2020c;b), depicted in Figure 2. Positive pairs for medical time-series are constructed using extensions of existing data augmentation techniques described in Cheng et al. (2020); Kiyasseh et al. (2020); Mohsenvand et al. (2020). Negative sampling is implemented via a momentum encoder (He et al., 2020). Finally, standard projection layers are incorporated as the final layers of the architecture (Chen et al., 2020a). As in He et al. (2020), we define an in-place momentum update. At each training step t, $f_m = (1 - \rho)f_e + \rho f_m$ and similarly $h_m = (1 - \rho)h_e + \rho h_m$ for hyperparameter $\rho \in (0, 1)$.

Patient-state encoder. Building on previous work in contrastive learning on time-series data, we use Temporal Convolutional Networks (TCN) (Bai et al., 2018). We slightly modify the original TCN architecture using layer normalization (Ba et al., 2016). We also add a dense layer after the TCN block to merge the static features from each patient into the final state representation. We refer to the Appendix B Figure 5 for a diagram of the architecture.

Data augmentations for online monitoring. We use channel dropout (Cheng et al., 2020) and Gaussian noise (Kiyasseh et al., 2020; Mohsenvand et al., 2020) to create positive pairs. In channel dropout, we randomly mask out a subset of variables while keeping the full time-series intact. Gaussian noise is simply the addition of independent Gaussian noise to each variable. Concerning the demographic vector in each sample, we only use random dropout. We did not use other channel augmentations from these approaches due to their specificity to EEG/ECG.

Using temporal augmentations from previous work is not straight forward. As explained in Section 4.1, contrary to EEG/ECG, large history overlaps exist between samples from adjacent states of a patient stay. Therefore randomly shifting or cropping the time-series component of these samples can lead to identical augmented views. Inspired by Cheng et al. (2020)'s work, we introduce two augmentations that never alter the last step of a time-series, preserving its singularity in the online monitoring context. First, history cutout, which masks out a short window from the time-series but excludes the last step from the possible candidates. Second, history crop, which crops a time-series along the temporal axis. However, we only crop from the past. More details about the data augmentations can be found in Appendix B.

5. Experimental Setup

5.1. Datasets

To benchmark the proposed method and allow further work to compare to our results, we selected two well-known EHR datasets that are openly available and for which online monitoring tasks exist.

MIMIC-III Benchmark. The MIMIC-III dataset (Johnson et al., 2016) is the most commonly used dataset for tasks related to EHR data. However, as noted by Bellamy et al. (2020), many previous approaches using it, applied their custom-built pre-processing pipeline and selection of variables, and thus making a comparison across methods impossible. To allow further comparison to our work, we used the pre-processed version of the dataset by Harutyunyan et al. (2019), referred to as "MIMIC-III Benchmark". Among the four tasks they define, we used the two with hourly labeling for each patient stay, *Decompensation* and *Length-of-stay* predictions.

Decompensation is a binary classification task which aims to predict whether a patient at time t, is going to pass away in the upcoming 24h. This task is highly unbalanced. Thus, as in Harutyunyan et al. (2019), we evaluate it with AUROC and AUPRC metrics.

For the *Length-of-stay* task, we aim to predict the remaining time the patient will stay in the ICU. This task is a regression problem. However, due to the heavy-tailed distribution of labels, Harutyunyan et al. (2019) frame it as a 10-way classification where each class is a binned duration of time. To evaluate the method, like them, we use linear weighted Cohen's Kappa. The score is between -1 and 1, 0 corresponding to random predictions.

The benchmark dataset extracted from MIMIC-III by Harutyunyan et al. (2019) contains more than 50,000 stays across 38,000 distinct patients for which 17 measurements are provided. More details about the dataset can be found in Appendix C.

Physionet 2019. This dataset originates from a challenge on the early detection of sepsis from clinical data (Reyna et al., 2019). It contains more than 40,000 patients from two hospitals, for which a total of 40 variables are available. Because the original test set was not available, we used the splits of (Horn et al., 2020) on the openly available sub-set of patients.

The task is to hourly predict sepsis onset occurring within the next 6h to 12h. To evaluate performances, we also used the novel "Utility" metric introduced by the authors of the challenge (Reyna et al., 2019). Compared to AUROC and AUPRC, this metric is more clinically relevant as it penalizes differently false predictions depending on their

relative temporal distance to a sepsis event. A score of zero corresponds to a classifier predicting no sepsis event. For a perfect classifier, the maximum score is one.

Table 1. Other contrastive methods as instances of our framework.

Method	α	w	$n(\cdot,\cdot)$
CL	1.0	0	n_w
SACL (Cheng et al., 2020)	0.0	$+\infty$	n_w
CLOCS (Kiyasseh et al., 2020)	1.0	$+\infty$	n_w
SCL (Khosla et al., 2020)	1.0	N.A	n_Y

5.2. Baselines

To achieve a fair comparison, we compare all approaches using the same encoder detailed in Section 5.3. Like others, we compare our method to the so-called "End-to-end" baseline. It consists of training the identical architecture in a supervised manner using downstream task labels.

We also compare our work to existing contrastive approaches reported in Table 1. All of them fall under different settings of our general framework. Regular CL, as a direct adaptation of Chen et al. (2020c;b) work, can be reproduced using a window size $w = 0$ and only $\mathcal{L}^{NA}$. Specific methods for medical time-series, SACL "Subject-specific" sampling method (Cheng et al., 2020) and CLOCS (Kiyasseh et al., 2020), are equivalent to using $w = +\infty$, with respectively only $\mathcal{L}^{ND}$ or $\mathcal{L}^{NA}$. Finally, SCL (Khosla et al., 2020) is similar to CLOCS, except its neighborhood function is defined based on the downstream task labels, making the method supervised.

We also compare the unsupervised version of our framework, using n_w, to Seq2Seq auto-encoders. First, an auto-encoder (Seq2Seq-AE), trained to minimize the mean square error (MSE) over the input time segment and static vector. Then, a forecasting one (Seq2Seq-AE-forecast), which defines the reconstruction loss over the consecutive time segment of the same length.

5.3. Implementation

Pre-processing. First, for each dataset, we re-sampled each stay to hourly resolution and filled missing values with forward-filling imputation. We then applied standard scaling to each non-categorical variable based on the training set statistics. We also one-hot encoded the remaining categorical variables. Finally, we zero-imputed (corresponding to the mean of the training set after scaling) the remaining missing values after the forward-filling imputation. We used a maximum history length t_h of 48 hours. We pre-padded shorter time-series. This pre-processing leads to input of size 48×42 for MIMIC-III Benchmark tasks and 48×40 for the Physionet 2019 one.

Architecture. From architecture searches (Appendix D) for the end-to-end baseline, we used the common encoder depicted in Figure 5, for both data sets. We use a convolution kernel of size 2 and 64 filters. To obtain a receptive field of at least 48 h, we stack five dilated causal convolution blocks. The final embedding dimension after incorporating the static feature is 64.

Unsupervised training parameters. We trained all unsupervised methods for 25k steps with a batch size of 2048. We used an Adam optimizer with a linear warm-up between 1e-5 and 1e-3 for 2.5k steps followed by cosine decay schedule as introduced by Chen et al. (2020a). We selected the common contrastive parameters from performances on the validation set for $\mathcal{L}^{CL}$ objective. More details can be found in Appendix D. We used a temperature of 0.1, a queue of size 65536, and an embedding size of 64 for all tasks. We set the momentum to 0.999 for MIMIC-III Benchmark tasks and 0.99 for Physionet 2019. Concerning parameters specific to our method, for NCL(n_w) we chose $\alpha = 0.3$ and $w = 16$ on MIMIC-III Benchmark and $\alpha = 0.4$ and $w = 12$ on Physionet 2019. For NCL(n_Y), we use $\alpha = 0.9$ for all tasks. These parameters were selected using grid searches reported in Appendix D. For auto-encoding methods, we used a decoder with a mirrored architecture to the common encoder. However, we did not normalize the representations to the unit sphere.

Model evaluation parameters. We evaluated all representation learning methods on a frozen representation. As discussed in Section 6 we used two different classification heads, a linear and a non-linear MLP. We used early stopping on validation set loss and an Adam optimizer. The learning rate was set to 1e-4 for all tasks and classification heads on the MIMIC-III benchmark dataset. For Physionet 2019 we used a learning rate of 1e-4 for linear classification and 5e-5 for the MLP head. For the end-to-end baseline, which trains the encoder and classification head simultaneously, we used a smaller learning rate of 1e-5.

6. Results

Before discussing results individually, one general thing to note is that end-to-end baselines are competitive with previous work. On the MIMIC-III Benchmark, it performs on par with the proposed methods by Harutyunyan et al. (2019). Similarly, for the same splitting of Physionet 2019, it outperforms all models from Horn et al. (2020).

NCL(n_w) closes the gap to end-to-end training. In the unsupervised setting, from Table 2, we observe that NCL(n_w), with an MLP head, is the only method to beat end-to-end training on both metrics for the *Decompensation* task. Also, by performing on par with the best unsupervised

Table 2. Results on the MIMIC-III Benchmark dataset. (Top rows) Unsupervised methods; (Bottom rows) Supervised methods. All scores are averaged over 20 runs such that the reported score is of the form $mean \pm std$. In bold are the methods within one standard deviation of best one for each setting. Evaluation metrics were scaled to 100 for readability purposes. (D) and (L) stands for Decompensation and Length-of-Stay indicating which labels were used to train the representation. To get competitive results we had to froze the projector for SCL.

Task	Decompensation				Length-of-stay	
Metric	AUPRC (in %)		AUROC (in %)		Kappa ($\times 100$)	
Head	Linear	MLP	Linear	MLP	Linear	MLP
Seq2-Seq-AE	17.5 ± 1.3	19.3 ± 1.5	84.8 ± 0.6	86.8 ± 0.4	38.2 ± 0.6	41.6 ± 0.3
Seq2Seq-AE-forecast	24.7 ± 1.3	28.7 ± 1.1	87.7 ± 0.4	89.7 ± 0.2	40.3 ± 0.3	42.2 ± 0.3
CL	31.0 ± 0.6	34.7 ± 0.4	88.3 ± 0.3	90.3 ± 0.2	40.4 ± 0.2	$\mathbf{43.2} \pm 0.2$
SACL (Cheng et al., 2020)	18.4 ± 1.9	29.3 ± 0.9	81.8 ± 1.3	87.5 ± 0.4	32.6 ± 2.0	40.1 ± 0.5
CLOCS (Kiyasseh et al., 2020)	29.9 ± 0.7	32.2 ± 0.8	89.5 ± 0.3	90.5 ± 0.2	41.7 ± 0.2	$\mathbf{43.0} \pm 0.2$
NCL(n_w) (Ours)	31.2 ± 0.5	$\mathbf{35.1} \pm 0.4$	88.9 ± 0.3	$\mathbf{90.8} \pm 0.2$	40.5 ± 0.3	$\mathbf{43.2} \pm 0.2$
End-to-End	34.3 ± 1.1	34.2 ± 0.6	90.6 ± 0.3	90.6 ± 0.2	43.3 ± 0.2	43.4 ± 0.2
SCL (D) (Khosla et al., 2020)	32.1 ± 0.9	31.9 ± 1.1	89.9 ± 0.3	89.5 ± 0.3	35.9 ± 0.7	40.2 ± 0.4
SCL (L) (Khosla et al., 2020)	30.6 ± 1.6	31.3 ± 0.8	86.1 ± 1.0	88.7 ± 0.4	41.3 ± 0.7	41.8 ± 0.4
NCL(n_Y) (D) (Ours)	$\mathbf{37.0} \pm 0.6$	$\mathbf{37.1} \pm 0.7$	90.3 ± 0.2	$\mathbf{90.9} \pm 0.1$	40.8 ± 0.3	43.3 ± 0.2
NCL(n_Y) (L) (Ours)	33.5 ± 1.0	36.0 ± 0.6	88.2 ± 0.5	90.5 ± 0.2	$\mathbf{43.7} \pm 0.2$	$\mathbf{43.8} \pm 0.3$

Table 3. Results on the Physionet 2019 dataset. (Top rows) Unsupervised methods; (Bottom rows) Supervised methods. All scores are averaged over 20 runs such that the reported score is of the form $mean \pm std$. In bold are the methods within one standard deviation of best one for each setting. Evaluation metrics were scaled to 100 for readability purposes.

Task	Sepsis onset prediction					
Metric	AUPRC (in %)		AUROC (in %)		Utility ($\times 100$)	
Head	Linear	MLP	Linear	MLP	Linear	MLP
Seq2-Seq-AE	7.0 ± 0.3	7.8 ± 0.4	77.1 ± 0.5	78.1 ± 0.6	26.8 ± 1.0	27.2 ± 1.0
Seq2-Seq-AE-forecast	6.6 ± 0.3	7.3 ± 0.3	75.8 ± 0.9	76.9 ± 0.5	23.5 ± 1.5	23.8 ± 1.2
CL	7.9 ± 0.4	$\mathbf{9.5} \pm 0.4$	78.2 ± 0.3	80.2 ± 0.4	26.2 ± 0.8	$\mathbf{29.7} \pm 1.0$
SACL (Cheng et al., 2020)	6.5 ± 0.3	7.6 ± 0.3	73.0 ± 1.2	75.3 ± 0.8	20.5 ± 2.5	24.2 ± 1.1
CLOCS (Kiyasseh et al., 2020)	7.1 ± 0.5	7.3 ± 0.4	77.2 ± 0.5	78.8 ± 0.4	23.0 ± 1.1	25.8 ± 0.9
NCL(n_w) (Ours)	8.2 ± 0.4	$\mathbf{9.3} \pm 0.5$	78.8 ± 0.3	$\mathbf{80.7} \pm 0.3$	27.2 ± 1.0	$\mathbf{30.2} \pm 1.0$
End-to-End	7.6 ± 0.2	8.1 ± 0.4	78.9 ± 0.3	78.8 ± 0.4	27.9 ± 0.8	27.5 ± 1.0
SCL (Khosla et al., 2020)	6.7 ± 0.6	6.0 ± 0.5	73.1 ± 1.7	70.0 ± 1.9	20.2 ± 2.7	20.6 ± 1.7
NCL(n_Y) (Ours)	$\mathbf{10.0} \pm 0.5$	$\mathbf{10.1} \pm 0.3$	80.3 ± 0.4	$\mathbf{80.8} \pm 0.2$	$\mathbf{32.6} \pm 1.0$	$\mathbf{31.9} \pm 0.9$

methods on *Length-of-stay* predictions, despite its additional parameters, NCL(n_w) is not task-specific. Moreover, as shown in Figure 3, using n_w alone is not sufficient. When not coupled to $\mathcal{L}^{\text{NCL}}$, performance significantly decreases.

While CLOCS exhibits good performance on the MIMIC-III Benchmark, it fails to do so for *Sepsis* predictions (Table 3). Furthermore, for *Sepsis* predictions, among the contrastive methods designed to tackle patient-induced heterogeneity, NCL(n_w) is the only one that improves performance over end-to-end training. However, CL achieving similar perfor-

mances as our method on the task suggests some limitations inherent to n_w.

NCL(n_Y) significantly improves over SCL. In the supervised setting, SCL has shown to be highly unstable to train. As shown in Table 2 and 3, SCL fails to learn a good representation. It has the lowest performance among all methods, both supervised and unsupervised.

We believe the low performance of SCL is due to highly similar time-series segments that can have different labels.

Table 4. Results for the limited-labeled data scenario on the Decompensation task. Each method was run on 4 random splits of the training data. Splitting was done at a patient level and splits were stratified to preserve the label prevalence.

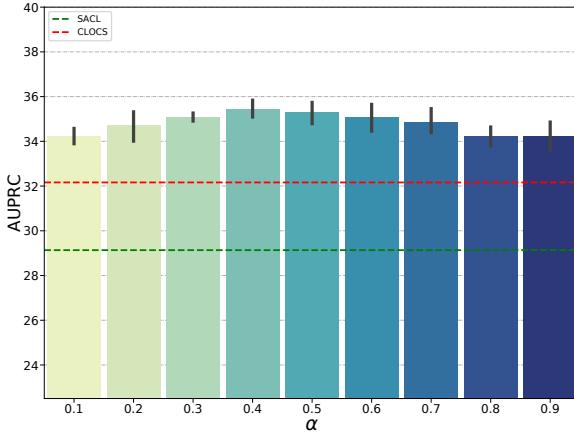| Task	Decompensation					
Labels	1 %		10 %		50 %	
Metric	AUPRC (in %)	AUROC (in %)	AUPRC (in %)	AUROC (in %)	AUPRC (in %)	AUROC (in %)
Seq2-Seq-AE	8.3 ± 1.4	79.4 ± 1.7	14.4 ± 1.0	84.7 ± 0.6	17.5 ± 1.3	86.1 ± 0.5
Seq2-Seq-AE-forecast	12.1 ± 1.5	83.0 ± 1.3	21.7 ± 2.0	87.7 ± 0.4	26.5 ± 1.4	89.1 ± 0.3
CL	$\mathbf{21.6} \pm 4.0$	$\mathbf{84.9} \pm 1.0$	$\mathbf{30.6} \pm 0.9$	88.6 ± 0.3	$\mathbf{33.9} \pm 0.4$	89.8 ± 0.2
SACL	12.3 ± 3.0	77.1 ± 1.6	20.6 ± 1.5	83.6 ± 0.8	27.4 ± 1.1	86.6 ± 0.5
CLOCS	13.1 ± 3.3	83.7 ± 1.7	26.1 ± 1.8	$\mathbf{88.9} \pm 0.4$	31.2 ± 0.9	$\mathbf{90.1} \pm 0.2$
NCL(n_w) (Ours)	19.8 ± 3.5	$\mathbf{85.2} \pm 1.3$	29.8 ± 1.3	$\mathbf{89.1} \pm 0.3$	$\mathbf{34.1} \pm 0.6$	$\mathbf{90.3} \pm 0.2$
End-to-End	13.1 ± 2.9	82.6 ± 1.4	25.8 ± 1.5	88.2 ± 0.4	32.3 ± 0.6	89.9 ± 0.2

Figure 3. Performance on the *Decompensation* task for various values of α in NCL(n_w). Results are averaged over 5 runs and $w = 16$. We observe a trade-off in performances when varying aggregation as conjectured in Section 3.
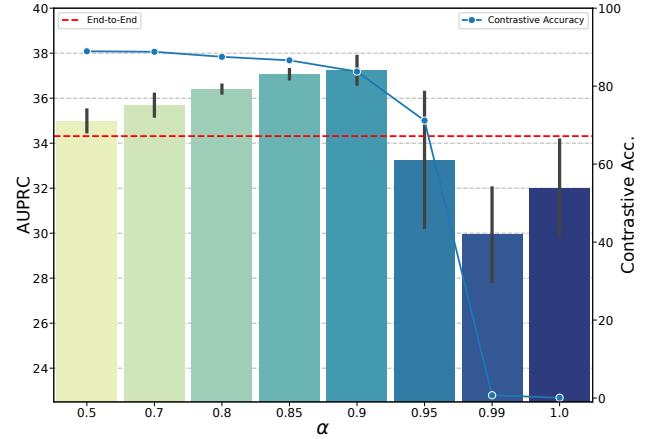

Figure 4. Performance on the *Decompensation* task for various values of α in NCL(n_Y). Results are averaged over 5 runs and trained with the *Decompensation* labels. In blue, we show the contrastive accuracy. For $\alpha > 0.95$ it drops because pre-text task becomes too hard explaining SCL's low performances.

It results in an excessively challenging pretext task known to hurt downstream performance (Tian et al., 2020). As shown in Figure 4, easing the pretext task, by lowering α, increases the model contrastive accuracy. On the downstream task, it yields a similar trade-off in performance as Tian et al. (2020) in the unsupervised case.

When using our objective $\mathcal{L}^{\text{NCL}}$ with the same neighborhood function n_Y as SCL, we observe a significant improvement on all tasks. On the MIMIC-III Benchmark (Table 2), when trained with the task labels, it outperforms all other methods, including SCL. More importantly, the features learned by NCL(n_Y) transfer much better to the other task. Even when trained with *Length-of-stay* labels, for the *Decompensation* task, our method still outperforms end-to-end training and all instances of SCL. Finally, as shown in Table 3, while

SCL fails on the sepsis task, NCL(n_Y) surpasses all other methods.

Frozen linear evaluation is limited for patient state representation. A common way to compare representation learning approaches in the literature has been to use a linear classifier on top of a learned representation. In our work, we show (Table 2, Table 3) that a non-linear classifier yields significantly better performances for all representation learning methods. However, end-to-end training does not benefit from a non-linear head. More than not capturing the full potential of a representation, linear evaluation can even be misleading. For instance, under linear evaluation, CLOCS outperforms our methods in AUROC for *Decompensation*. However, with a MLP head, even if both improve, their

relative ordering in performance changes.

Reducing amount of labels improves over supervised training. To evaluate unsupervised representations, it is common practice to compare them to their end-to-end counterparts while decreasing the amount of labeled data. To this end, we report results using fractions of *Decompensation* label amounts in Table 4. We show that $NCL(n_w)$ significantly outperforms end-to-end training and other patient-designed methods on both AUROC and AUPRC when the number of labeled patients is reduced. However, while our method stays competitive with regular CL in AUROC, it is not the case for AUPRC. Such a discrepancy between the two metrics suggests an increasing proportion of false positives when reducing labeled data for $NCL(n_w)$.

7. Conclusion

In this paper, our aim is to bring state-of-the-art contrastive learning methods one step closer to being applied to online patient state monitoring in the ICU. Our work addresses the domain heterogeneity problems inherent to this task by introducing a contrastive objective which encourages patient state representations to follow distributional assumptions dictated by prior knowledge.

By operating in conjunction with existing augmentation techniques, we are able to make contrastive learning a more expressive framework for working with challenging real-world datasets and incorporating adjacent information and prior knowledge.

When fully-unsupervised, our method shows competitive results over end-to-end training and previous self-supervised approaches for biosignals. Besides, when used in a supervised manner, our framework considerably improves over existing supervised contrastive learning methods.

In addition to the choice of data augmentations, our method depends strongly on the definition of the neighborhood function. Our binary neighborhood framework is the first step in encouraging higher-order behavior of contrastive learning representations via loss functions. We believe that relaxing this definition from being binary to categorical or continuous is a promising direction for future work.

8. Acknowledgements

References

Ba, J. L., Kiros, J. R., and Hinton, G. E. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

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

Banville, H., Chehab, O., Hyvarinen, A., Engemann, D., and Gramfort, A. Uncovering the structure of clinical eeg signals with self-supervised learning. *Journal of Neural Engineering*, 2020.

Bellamy, D., Celi, L., and Beam, A. L. Evaluating progress on machine learning for longitudinal electronic healthcare data. *arXiv preprint arXiv:2010.01149*, 2020.

Caron, M., Misra, I., Mairal, J., Goyal, P., Bojanowski, P., and Joulin, A. Unsupervised learning of visual features by contrasting cluster assignments. *arXiv preprint arXiv:2006.09882*, 2020.

Chen, T., Kornblith, S., Norouzi, M., and Hinton, G. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020a.

Chen, T., Kornblith, S., Swersky, K., Norouzi, M., and Hinton, G. Big self-supervised models are strong semi-supervised learners. *arXiv preprint arXiv:2006.10029*, 2020b.

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

Cheng, J. Y., Goh, H., Dogrusoz, K., Tuzel, O., and Azemi, E. Subject-aware contrastive learning for biosignals. *arXiv preprint arXiv:2007.04871*, 2020.

Darabi, S., Kachuee, M., and Sarrafzadeh, M. Unsupervised representation for ehr signals and codes as patient status vector. *arXiv preprint arXiv:1910.01803*, 2019.

Farshchian, A., Gallego, J. A., Cohen, J. P., Bengio, Y., Miller, L. E., and Solla, S. A. Adversarial domain adaptation for stable brain-machine interfaces. *arXiv preprint arXiv:1810.00045*, 2018.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. Data augmentation using synthetic data for time series classification with deep residual networks. *arXiv preprint arXiv:1808.02455*, 2018.

Franceschi, J.-Y., Dieuleveut, A., and Jaggi, M. Unsupervised scalable representation learning for multivariate time series. *arXiv preprint arXiv:1901.10738*, 2019.

Gutmann, M. and Hyvärinen, A. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pp. 297–304. JMLR Workshop and Conference Proceedings, 2010.

Harutyunyan, H., Khachatrian, H., Kale, D. C., Ver Steeg, G., and Galstyan, A. Multitask learning and benchmarking with clinical time series data. *Scientific data*, 6(1): 1–18, 2019.

He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9729–9738, 2020.

Horn, M., Moor, M., Bock, C., Rieck, B., and Borgwardt, K. Set functions for time series. In *International Conference on Machine Learning*, pp. 4353–4363. PMLR, 2020.

Hyland, S. L., Faltys, M., Hüser, M., Lyu, X., Gumbsch, T., Esteban, C., Bock, C., Horn, M., Moor, M., Rieck, B., et al. Early prediction of circulatory failure in the intensive care unit using machine learning. *Nature medicine*, 26(3):364–373, 2020.

Johnson, A. E., Pollard, T. J., Shen, L., Li-Wei, H. L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., and Mark, R. G. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3(1):1–9, 2016.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*, 2020.

Killian, T. W., Zhang, H., Subramanian, J., Fatemi, M., and Ghassemi, M. An empirical study of representation learning for reinforcement learning in healthcare. *arXiv preprint arXiv:2011.11235*, 2020.

Kiyasseh, D., Zhu, T., and Clifton, D. A. Clocs: Contrastive learning of cardiac signals. *arXiv preprint arXiv:2005.13249*, 2020.

Landi, I., Glicksberg, B. S., Lee, H.-C., Cherng, S., Landi, G., Danieletto, M., Dudley, J. T., Furlanello, C., and Miotto, R. Deep representation learning of electronic health records to unlock patient stratification at scale. *NPJ digital medicine*, 3(1):1–11, 2020.

Li, X., Xu, X., Xie, F., Xu, X., Sun, Y., Liu, X., Jia, X., Kang, Y., Xie, L., Wang, F., et al. A time-phased machine learning model for real-time prediction of sepsis in critical care. *Critical Care Medicine*, 48(10):e884–e888, 2020.

Lyu, X., Hueser, M., Hyland, S. L., Zerveas, G., and Raetsch, G. Improving clinical predictions through unsupervised time series representation learning. *arXiv preprint arXiv:1812.00490*, 2018.

McDermott, M., Nestor, B., Kim, E., Zhang, W., Goldenberg, A., Szolovits, P., and Ghassemi, M. A comprehensive evaluation of multi-task learning and multi-task pre-training on ehr time-series data. *arXiv preprint arXiv:2007.10185*, 2020.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

Miotto, R., Li, L., Kidd, B. A., and Dudley, J. T. Deep patient: an unsupervised representation to predict the future of patients from the electronic health records. *Scientific reports*, 6(1):1–10, 2016.

Misra, I. and Maaten, L. v. d. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6707–6717, 2020.

Mohsenvand, M. N., Izadi, M. R., and Maes, P. Contrastive representation learning for electroencephalogram classification. In *Machine Learning for Health*, pp. 238–253. PMLR, 2020.

Morioka, H., Kanemura, A., Hirayama, J.-i., Shikauchi, M., Ogawa, T., Ikeda, S., Kawanabe, M., and Ishii, S. Learning a common dictionary for subject-transfer decoding with resting calibration. *NeuroImage*, 111:167–178, 2015.

Özdenizci, O., Wang, Y., Koike-Akino, T., and Erdoğmuş, D. Learning invariant representations from eeg via adversarial inference. *IEEE access*, 8:27074–27085, 2020.

Reyna, M. A., Josef, C., Seyedi, S., Jeter, R., Shashikumar, S. P., Westover, M. B., Sharma, A., Nemati, S., and Clifford, G. D. Early prediction of sepsis from clinical data: the physionet/computing in cardiology challenge 2019. In *2019 Computing in Cardiology (CinC)*, pp. Page–1. IEEE, 2019.

Schwab, P., Mehrjou, A., Parbhoo, S., Celi, L. A., Hetzel, J., Hofer, M., Schölkopf, B., and Bauer, S. Real-time prediction of covid-19 related mortality using electronic health records. *arXiv preprint arXiv:2008.13412*, 2020.

Tian, Y., Sun, C., Poole, B., Krishnan, D., Schmid, C., and Isola, P. What makes for good views for contrastive learning. *arXiv preprint arXiv:2005.10243*, 2020.

Tomašev, N., Glorot, X., Rae, J. W., Zielinski, M., Askham, H., Saraiva, A., Mottram, A., Meyer, C., Ravuri, S., Protsyuk, I., et al. A clinically applicable approach to continuous prediction of future acute kidney injury. *Nature*, 572(7767):116–119, 2019.

Um, T. T., Pfister, F. M., Pichler, D., Endo, S., Lang, M., Hirche, S., Fietzek, U., and Kulić, D. Data augmentation of wearable sensor data for parkinson's disease monitoring using convolutional neural networks. In *Proceedings of the 19th ACM International Conference on Multimodal Interaction*, pp. 216–220, 2017.

Wang, T. and Isola, P. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pp. 9929–9939. PMLR, 2020.

A. Architecture

In this section, we expand on the details of our architecture. The full architecture of the encoder is depicted in Figure 5. For the non-linear projector and classifier, we used an inner layer with the same dimension as the representation size. Thus for all tasks, we used an inner dimension of 64. Because we deal with time-series, we used causal dilated convolutions to not break temporal ordering. We built our pipeline using `tensorflow 2.3` and `keras-tcn 3.3`.
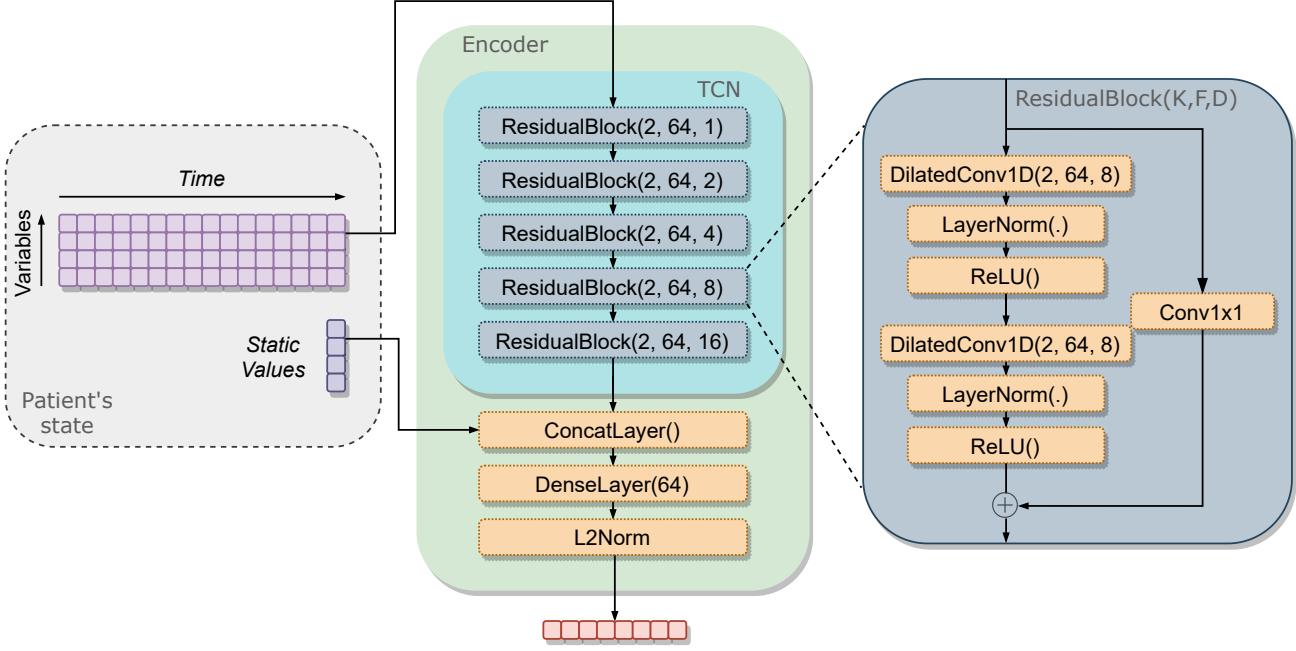

Figure 5. Encoder architecture we used for all methods. In the figure, K, F, and D represent respectively, the kernel size, number of filters, and dilation rate. We use a similar TCN block than the original paper (Bai et al., 2018) with the exception that we use layer normalization. We use a fully-connected layer to incorporate static features in the representation. Finally, we normalize this representation to the unit sphere as in He et al. (2020)

B. Data augmentation

In this section, we further expand on the data augmentations used in all contrastive methods. To choose each function's hyperparameters we performed a random search on the validation performance for both MIMIC-III Benchmark and Physionet 2019 for the regular CL method. As a result of the random search, we chose the following parameters.

1. **History Crop**: We apply a crop with a probability of 0.5 and minimum size of 50% of the initial sequence.

2. **History Cutout**: We apply time cutout of 8 steps with a probability of 0.8.

3. **Channel Dropout**: We mask out each channel randomly with a probability of 0.2

4. **Gaussian Noise**: We add random Gaussian noise to each variable independently with a standard deviation of 0.1

Also, we verify that composing augmentations (Chen et al., 2020a) improves performances. We find, as in Cheng et al. (2020) and (Kiyasseh et al., 2020), that composing temporal and spatial augmentations yields the best performances as shown in Figure 6. It obtains lower performance than composing all transformations, which achieves an AUPRC of 35.5 on validation for the 5 same seeds. Therefore, we applied these four augmentations sequentially to both branches of the pipeline for all contrastive methods.

C. Data sets

In this section we expand further on the datasets we performed experiments on.
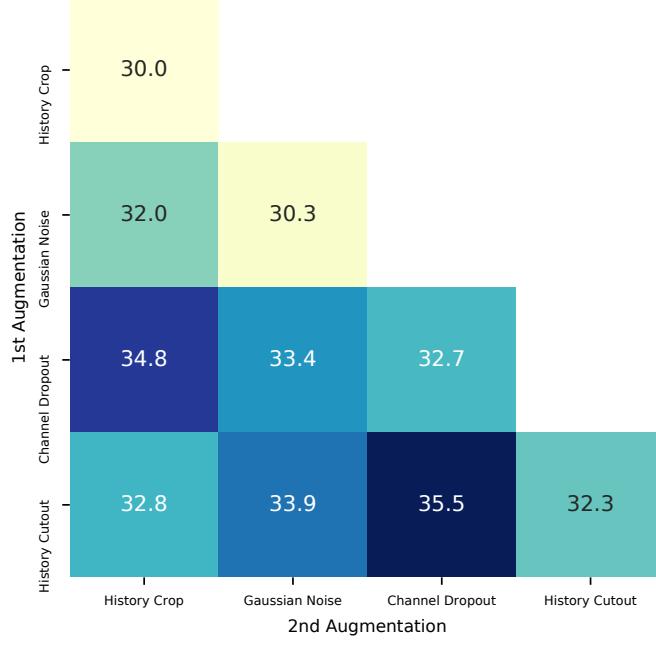
Figure 6. Comparison of performance between different choices of augmentation. Result are reported on validation set AUPRC for the decompensation task and are averaged over 5 seeds.

C.1. MIMIC-III Benchmark

As shown in Table 6, MIMIC-III Benchmark provides 17 measurements in addition to the time since admission. After one-hot encoding of the categorical features, we obtain an input dimension of 42.

In Table 5, we detail the splitting and prevalence of the dataset. We observe that, compared to Physionet 2019, the length of patient stays are significantly greater. Moreover, we also observe that decompensation is a highly unbalanced task.
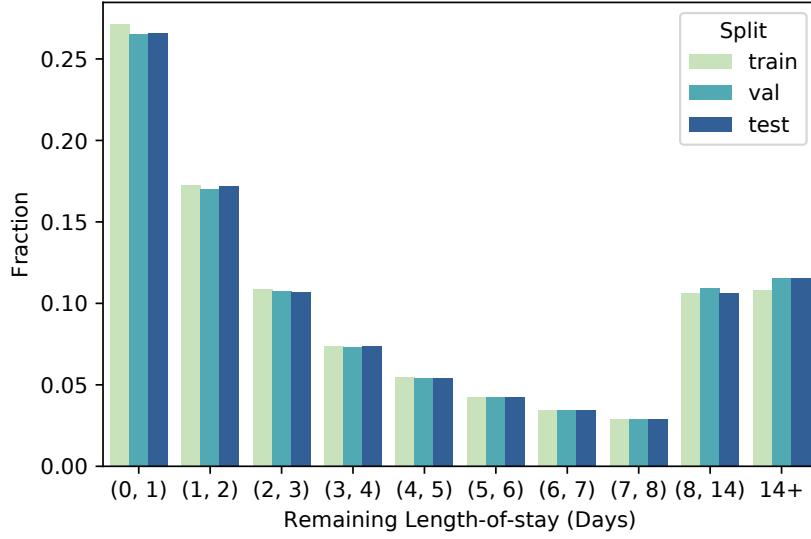

Figure 7. Prevalence of each temporal bin used in the *Length-of-stay* task. We used the same bins as (Harutyunyan et al., 2019).

Table 5. Number of patients and samples in the full data-set as well as for individual predictive tasks.

MIMIC-III	Number of patients		
	Train	Test	Val
	29250	6281	6371
Length of stay	**Number of samples**		
	Train	Test	Val
	2,586,619	563,742	572,032
Decompensation	**Number of samples**		
	Train	Test	Val
	2,377,738	523,200	530,638
	Number of positives		
	Train	Test	Val
	49,260	9,683	11,752

Table 6. Measurements recorded and re-sampled hourly in the MIMIC-III benchmark dataset. BP: Blood pressure, MAP: Mean arterial pressure, FiO_2: Fraction of inspired oxygen. GCS: Glasgow Coma Scale. SpO_2: Pulse oxygen saturation.

Measurement	Type
Time since admission	Continuous
Height	Static (Continuous)
Capillary refill rate	Categorical
GCS eye opening	Categorical
GCS motor response	Categorical
GCS verbal response	Categorical
GCS total	Categorical
Diastolic BP	Continuous
FiO_2	Continuous
Glucose	Continuous
Heart Rate	Continuous
MAP	Continuous
SpO_2	Continuous
Respiratory rate	Continuous
Systolic BP	Continuous
Temperature	Continuous
Weight	Continuous
pH	Continuous

C.2. Physionet 2019

As shown in Table 7 Physionet 2019 provides 40 measurements. As all categorical features are binary, the final input dimension is 40 as well. In Table 8, we detail the splitting and prevalence of the dataset. We, once again, highlight the very low prevalence of positive labels.

Table 7. Measurement recorded and re-sampled hourly for Physionet 2019. BP: Blood pressure, MAP: Mean arterial pressure, FiO_2: Fraction of inspired oxygen. $PaCO_2$: Partial pressure of carbon dioxide from arterial blood. SaO_2: Oxygen saturation from arterial blood. SpO_2: Pulse oxygen saturation.

Measurement	Type	Measurement	Type
Time since admission (ICU)	Continuous	SaO_2	Continuous
Age	Static (Continuous)	Aspartate transaminase	Continuous
Gender	Static (Categorical)	Blood urea nitrogen	Continuous
Hospital Admission Time	Static (Continuous)	Alkaline phosphatase	Continuous
ICU Unit 1	Static (Categorical)	Calcium	Continuous
ICU Unit 2	Static (Categorical)	Chloride	Continuous
Heart rate	Continuous	Creatinine	Continuous
SpO_2	Continuous	Bilirubin direct	Continuous
Temperature	Continuous	Total bilirubin	Continuous
Systolic BP	Continuous	Serum glucose	Continuous
MAP	Continuous	Lactic acid	Continuous
Diastolic BP	Continuous	Troponin I	Continuous
Respiratory rate	Continuous	Hematocrit	Continuous
End tidal carbon dioxide	Continuous	Hemoglobin	Continuous
Excess Bicarbonate	Continuous	Partial Thromboplastin time	Continuous
Bicarbonate	Continuous	Leukocyte count	Continuous
FiO_2	Continuous	Fibrinogen	Continuous
$PaCO_2$	Continuous	Platelets	Continuous

Table 8. Description of Physionet 2019 statistics by patient and sample.

Physionet 2019	**Number of patients**		
	Train	Test	Val
	25,813	8,066	6,454
Sepsis onset	**Number of samples**		
	Train	Test	Val
	992,732	312,078	247,283
	Number of positives		
	Train	Test	Val
	17,891	5,550	4,475

D. Hyperparameter selection

We tuned all existing hyperparameters over validation performances. For MIMIC-III, we used AUPRC on *Decompensation* task as a reference. For Physionet 2019 we used the Utility metric from (Reyna et al., 2019).

D.1. Architecture parameters

The main hyperparameters of the TCN architecture are the kernel size and the number of filters. We tuned these parameters on the End-to-end model and then used them for all other methods.

Kernel Size	AUPRC (Validation set)
2	**37.0** $\pm$ 0.5
4	36.7 $\pm$ 0.4
8	35.7 $\pm$ 0.6

Number of filters	AUPRC (Validation set)
16	37.1 $\pm$ 0.4
32	37.0 $\pm$ 0.5
64	**37.3** $\pm$ 0.5
128	37.1 $\pm$ 0.5
256	36.8 $\pm$ 1.0
512	35.7 $\pm$ 2.0

Table 9. (a) Impact of the kernel size parameter on the validation AUPRC metric for end-to-end training on MIMIC-III decompensation task. Results are averaged over 5 seeds and number of filters was set to 32, (b) Impact of the number of filters on the validation AUPRC metric for end-to-end training on the MIMIC-III decompensation task. Results are averaged over 5 seeds and kernel size was set to 2

Kernel Size	Utility (Validation set)
2	**28.7** $\pm$ 0.7
4	28.0 $\pm$ 1.1
8	**29.1** $\pm$ **1.6**

Number of filters	Utility (Validation set)
16	27.8 $\pm$ 1.4
32	28.8 $\pm$ 0.6
64	**29.0** $\pm$ 0.8
128	28.8 $\pm$ 1.3
256	26.8 $\pm$ 3.1

Table 10. (a) Impact of the kernel size parameter on the validation Utility metric for end-to-end training on Physionet 2019. Results are averaged over 5 seeds and the number of filters was set to 32., (b) Impact of the number of filter on the validation Utility metric for end-to-end training on Physionet 2019. Results are averaged over 5 seeds and the kernel size was set to 2

D.2. Contrastive parameters

The two main contrastive parameters shared across methods are the momentum ρ and the temperature τ. For a fair comparison, we used the same values for these parameters based on the performance of regular CL as shown in Figure 8 and 9.
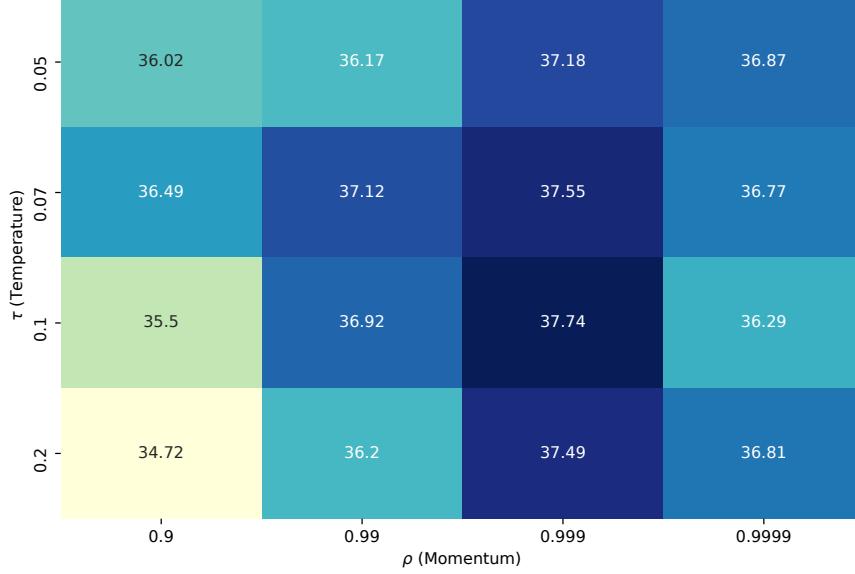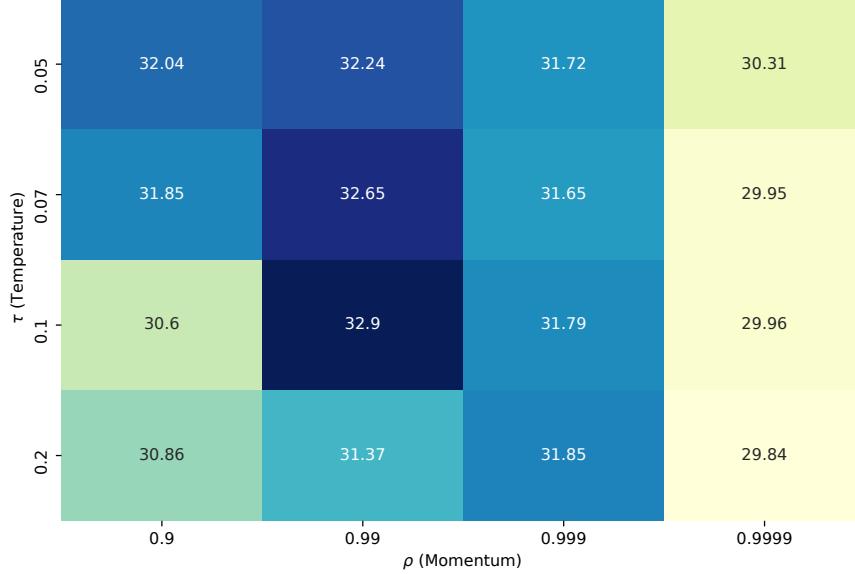

Figure 8. Grid search over τ (temperature) and ρ (momentum) for regular Contrastive Learning method on MIMIC-III. Here result are averaged over 5 runs. Reported metric is AUPRC on validation set for *Decompensation* task.

Figure 9. Grid search over τ (temperature) and ρ (momentum) for regular Contrastive Learning method on Physionet 2019. Here result are averaged over 5 runs. Reported metric is Utility on validation set for sepsis task.

D.3. Neighborhood parameters

As shown in Figures 13 and 14, we select the specific parameters to n_w with a grid search over 5 runs. If parameters yielding good performance are stable for MIMIC-III, we found that performance on Utility metric varied significantly for Physionet

2019. We believe a reason for that is the fact these metrics depend on a threshold for making a prediction. Thus, contrary to AUROC or AUPRC, in addition to evaluating the model performances, it also evaluates its calibration.

As showed in Figures 10, 11 and 12 we selected α for $NCL(n_Y)$ on validation set performance. We observe that for all tasks, taking $\alpha = 0.9$ yields the best performance on the training task and in transfer learning.
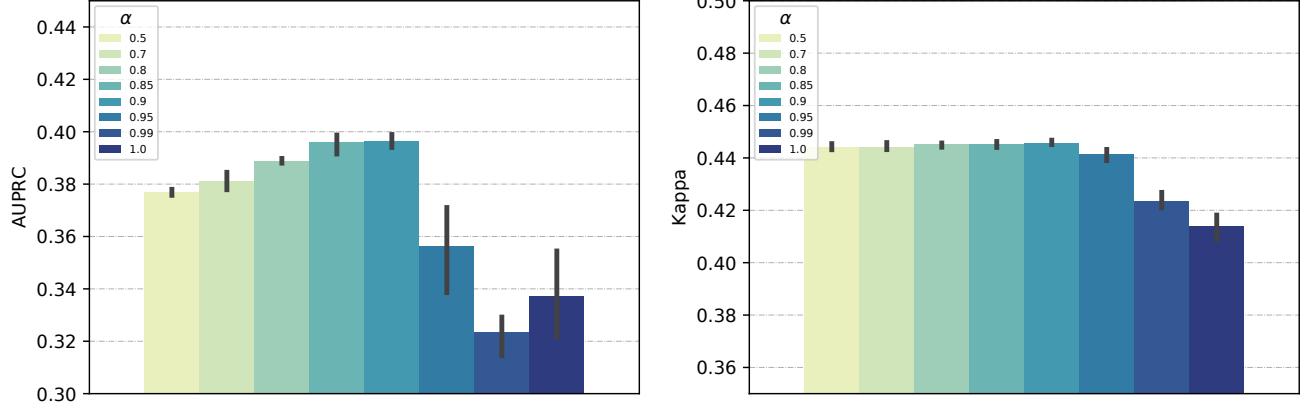

Figure 10. Parameter search over α for $NCL(n_Y)$ method on MIMIC-III Benchmark when trained using *Decompensation* labels. Here result are averaged over 5 runs. Reported metric is AUPRC (for *Decompensation*) and Kappa (for *Length-of-stay*)on validation set.
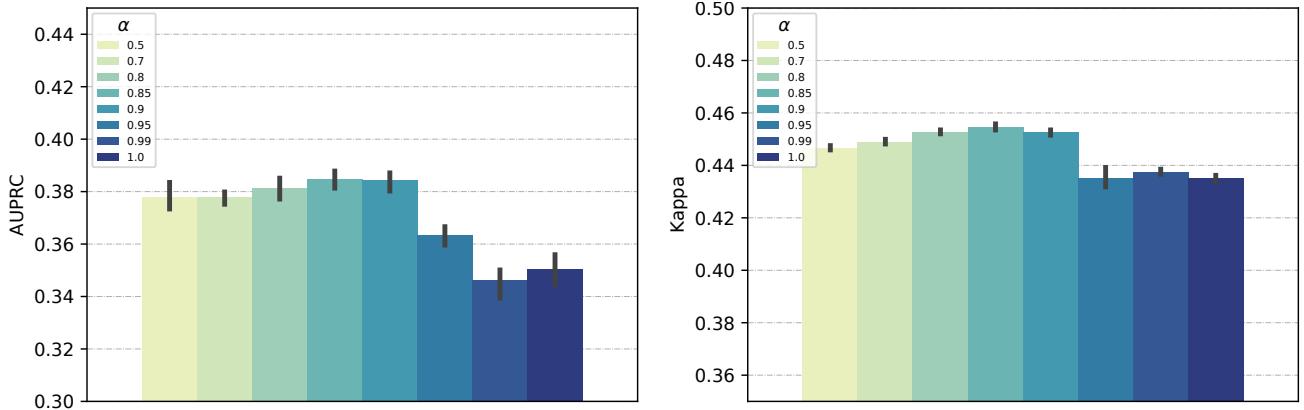

Figure 11. Parameter search over α for $NCL(n_Y)$ method on MIMIC-III Benchmark when trained using *Length-of-stay* labels. Here result are averaged over 5 runs. Reported metric is AUPRC (for *Decompensation*) and Kappa (for *Length-of-stay*)on validation set.
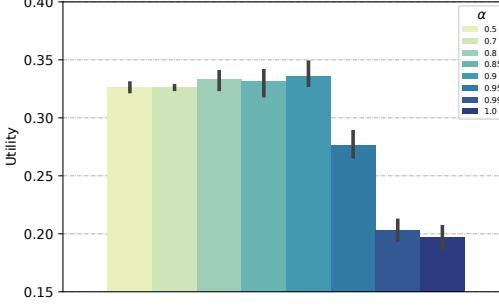
Figure 12. Parameter search over α for NCL(n_Y) method on Physionet 2019. Here result are averaged over 5 runs. Reported metric is Utility on validation set for sepsis task.

Figure 13. Grid search over w and α for NCL(n_w) method on MIMIC-III Benchmark. Here result are averaged over 5 runs. Reported metric is AUPRC on validation set for decompensation task.

Figure 14. Grid search over w and α for NCL(n_w) method on Physionet 2019. Here result are averaged over 5 runs. Reported metric is Utility on validation set for sepsis task.

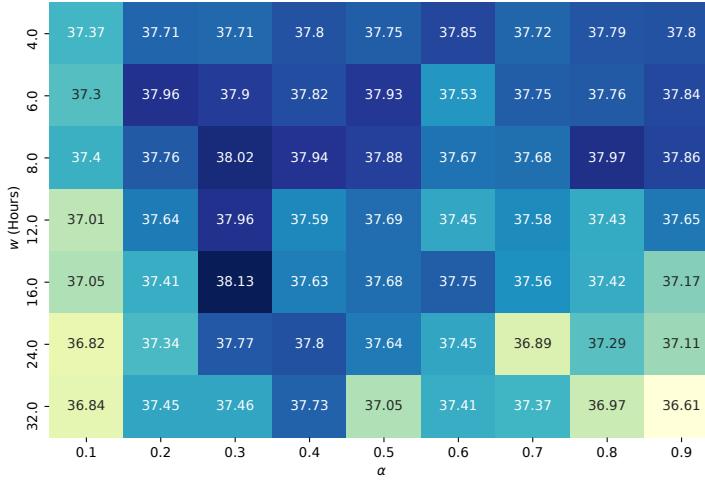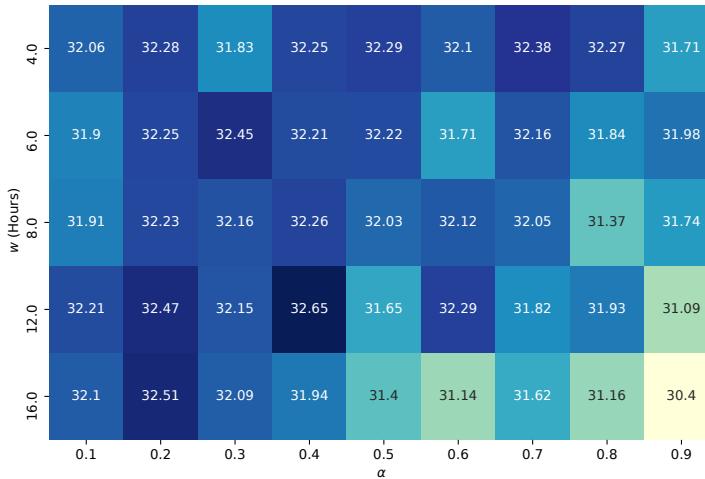# E. Supplementary Results

E.1. Semi-supervised neighborhood

We explored another neighborhood possibility for our supervised approach, as the intersection of N_w and N_Y, called $NCL(n_{w \cap Y})$. We make two observations from the results in Table 11. First, regardless of the label used for training, results were similar on all tasks and competitive with end-to-end. Second, we managed to achieve a stable training even though $\alpha = 1.0$ by considering as positive the sample with the same label and close temporally. However, we under-perform compared to $NCL(n_Y)$, suggesting that using the $\mathcal{L}^{NCL}$ objective is a better alternative in this case.

Table 11. Supplementary results on MIMIC-III when using other neighborhood function for supervised approach $NCL(n_{w \cap Y})$. (D) and (L) stands for Decompensation and Length-of-Stay indicating which labels were used at training. For $NCL(n_{w \cap Y})$ we used a trade-off parameter $\alpha = 1.0$

Task	Decompensation				Length-of-stay	
Metric	AUPRC		AUROC		Kappa	
Head	Linear	MLP	Linear	MLP	Linear	MLP
End-to-End	34.3 ± 1.1	34.2 ± 0.6	90.6 ± 0.3	90.6 ± 0.2	43.3 ± 0.2	$\mathbf{43.4} \pm 0.2$
$NCL(n_{w \cap Y})$ (D) (Ours)	31.3 ± 0.7	34.4 ± 0.6	89.4 ± 0.3	90.7 ± 0.1	40.8 ± 0.3	43.2 ± 0.2
$NCL(n_{w \cap Y})$ (L) (Ours)	31.2 ± 0.5	34.0 ± 0.4	89.2 ± 0.2	90.6 ± 0.1	40.6 ± 0.2	43.2 ± 0.2
$NCL(n_Y)$ (D) (Ours)	$\mathbf{37.0} \pm 0.6$	$\mathbf{37.1} \pm 0.7$	90.3 ± 0.2	$\mathbf{90.9} \pm 0.1$	40.8 ± 0.3	43.3 ± 0.2
$NCL(n_Y)$ (L) (Ours)	33.5 ± 1.0	36.0 ± 0.6	88.2 ± 0.5	90.5 ± 0.2	$\mathbf{43.7} \pm 0.2$	$\mathbf{43.8} \pm 0.3$

E.2. Fine-tuning representation

The preferred approach to compare representations is to use a frozen classifier. Contrary to fine-tuning, it preserves what was learned at the training step, allowing a fair comparison. However, if one is interested in the absolute performance on a downstream task, fine-tuning the representation encoder with the classification head usually yields better results. We show in Table 12 that for MIMIC-III, we observe this effect by further improving our unsupervised method. However, as shown in Table 13, on Physionet 2019 where our unsupervised method already improved over end-to-end training, performances are degraded. Moreover, we observe that the existing gap between classification heads disappears when fine-tuning. It highlights that fine-tuning shouldn't be used to compare representations learning approaches.

Table 12. Fine-tuning results for MIMIC-III. The results are averaged over the same 20 runs as frozen evaluation.

Task	Decompensation				Length-of-stay	
Metric	AUPRC		AUROC		Kappa	
Head	Linear	MLP	Linear	MLP	Linear	MLP
End-to-End	34.3 ± 1.1	34.2 ± 0.6	90.6 ± 0.3	90.6 ± 0.2	43.3 ± 0.2	43.4 ± 0.2
$NCL(n_w)$ (Ours)	36.7 ± 0.5	36.6 ± 0.4	91.1 ± 0.1	91.3 ± 0.2	43.7 ± 0.3	43.9 ± 0.3
$NCL(n_y)$ (D) (Ours)	36.7 ± 0.7	37.1 ± 0.7	90.7 ± 0.2	90.9 ± 0.1	44.0 ± 0.2	44.0 ± 0.3
$NCL(n_y)$ (L) (Ours)	34.8 ± 1.1	34.7 ± 1.0	90.2 ± 0.2	90.3 ± 0.3	42.5 ± 0.3	42.8 ± 0.3

Table 13. Fine-tuning results for Physionet 2019. The results are averaged over the same 20 runs as frozen evaluation.

Task	Sepsis					
Metric	AUPRC		AUROC		Utility	
Head	Linear	MLP	Linear	MLP	Linear	MLP
End-to-End	7.6 ± 0.2	8.1 ± 0.4	78.9 ± 0.3	78.8 ± 0.4	27.9 ± 0.8	27.5 ± 1.0
NCL(n_w) (Ours)	8.8 ± 0.4	8.9 ± 0.4	80.6 ± 0.4	80.7 ± 0.3	30.2 ± 0.9	30.3 ± 0.9
NCL(n_y) (Ours)	8.9 ± 0.4	9.5 ± 0.4	80.6 ± 0.3	80.9 ± 0.2	30.5 ± 0.7	31.6 ± 0.7

E.3. Visualizing Aggregation Impact

In Figure 15, using T-SNE plots we highlight that by increasing α in $\mathcal{L}^{\text{NCL}}$, we gradually increase aggregation among neighbors. As conjectured, using only $\mathcal{L}^{\text{NA}}$ or $\mathcal{L}^{\text{ND}}$. yields either a collapsed representation or a patient-independent representation.
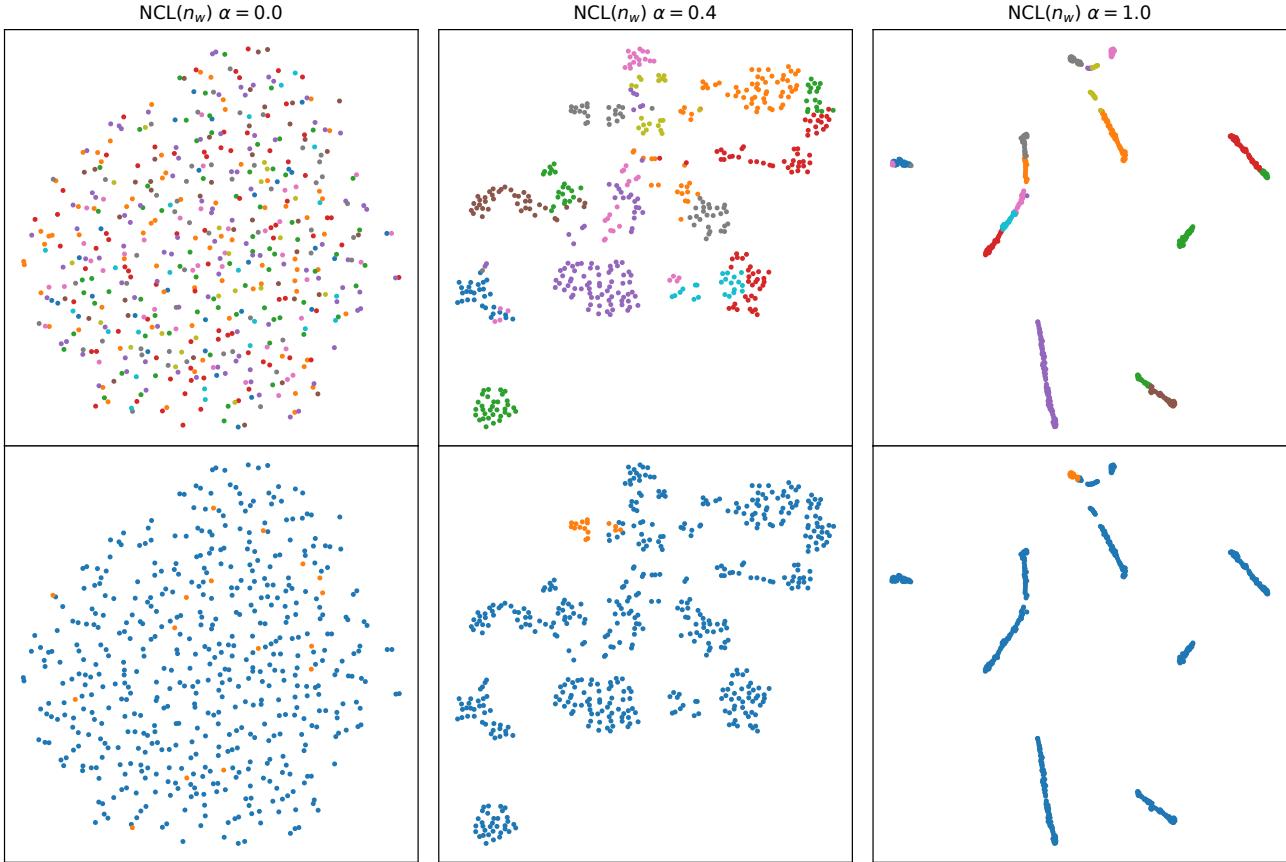

Figure 15. T-SNE plot (Mikolov et al., 2013) of learned representations on MIMIC-III Benchmark dataset for different values of α. (Top row) Labeled per patient. (Bottom row) Labeled with *Decompensation* task. We observe that as conjectured a trade-off in neighbors aggregation is obtained where taking an intermediate value for α.