Temporal-Clustering Invariance in Irregular Healthcare Time Series

Mohammad Taha Bahadori
bahadorm@amazon.com
Amazon

Zachary C. Lipton*
zlipton@amazon.com
Amazon

ABSTRACT

Electronic records contain sequences of events, some of which take place all at once in a single visit, and others that are dispersed over multiple visits, each with a different timestamp. We postulate that fine temporal detail, e.g., whether a series of blood tests are completed at once or in rapid succession should not alter predictions based on this data. Motivated by this intuition, we propose models for analyzing sequences of multivariate clinical time series data that are invariant to this temporal clustering. We propose an efficient data augmentation technique that exploits the postulated temporal-clustering invariance to regularize deep neural networks optimized for several clinical prediction tasks. We introduce two techniques to temporally coarsen (downsample) irregular time series: (i) grouping the data points based on regularly-spaced timestamps; and (ii) clustering them, yielding irregularly-paced timestamps. Moreover, we propose a MultiResolution network with Shared Weights (MRSW), improving predictive accuracy by combining predictions based on inputs sequences transformed by different coarsening operators. Our experiments show that MRSW improves the mAP on the benchmark mortality prediction task from 51.53% to 53.92%, which is the new state of the art on the benchmark.

CCS CONCEPTS

• **Computing methodologies** → **Machine learning algorithms**; • **Applied computing** → **Health informatics**; • **Mathematics of computing** → **Time series analysis**.

KEYWORDS

Multivariate Time Series, Healthcare, Data Augmentation

ACM Reference Format:
Mohammad Taha Bahadori and Zachary C. Lipton. 2020. Temporal-Clustering Invariance in Irregular Healthcare Time Series. In *CHIL '20: ACM Conference on Health, Inference, and Learning, June 03–05, 2020, Toronto, ON*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

While common academic sequence learning challenges, e.g., in natural language processing and speech, typically consist of evenly-spaced inputs, in healthcare, we frequently encounter multivariate time series, where the variables are sampled irregularly and exhibit signicantly varying base frequencies. The most popular sequence learning algorithms, based on recurrent neural networks, act upon discretized time steps, with no agreed-upon out-of-the-box method for handling such irregularities. Thus, to date, deep learning researchers have relied on heuristics, such as ignoring the timestamp information [11], binning the data into uniformly-spaced time intervals [20, 34], and manual feature extraction [49].

Several principled approaches have been proposed, falling broadly into two categories: (i) methods that adapt classical methods naturally capable of handling irregularly sampled data, such as point process models [23, 53] and Gaussian processes [15, 32, 45, 46]—these works extend classical approaches with the goal of making their accuracy and scalability competitive with modern deep learning approaches; (ii) methods that adapt recurrent neural networks [1, 4, 7, 8, 35, 47, 54] or convolutional neural networks [30, 38, 42] to the irregular time series setting.

To date, few papers have studied what invariances might hold in irregular time series data, or mechanisms by which learning algorithms might exploit them. Suppose a patient visits a hospital for an emergency service and undergoes a set of operations. Had she visited a smaller clinic, she might have undergone only a subset of those and the rest would have been done at a later time in another clinic. The first process will result in a single event in the patient's record, whereas the second will result in two events. Similar patterns can also emerge owing to the timing of the admission, insurance policies, or other factors related to the patient's convenience. Ideally, a learning algorithm should be robust to the temporal clustering of events. However, determining the best way to exploit this structure is not straightforward, and discarding information can be problematic. For example, the frequency of visits is often informative about the severity of illness and the algorithms should be able to use it in the analysis.

As the main contribution of this work, we investigate the temporal-clustering properties of clinical time series data. To address the existence and usefulness of temporal-clustering invariance, we demonstrate data augmentation schemes to exploit it. We propose an efficient data augmentation operator that coarsens the time series through a stochastic clustering. The method consists of randomly merging adjacent events, merging those more closely-spaced in time with higher probability. Our experiments demonstrate that this augmentation technique, in combination with neural network classifiers, can yield both more accurate classification and greater robustness to variations in the temporal-clustering of the data. The data augmentation pipeline allows us to access representations of our inputs sequences at multiple resolutions throughout training. We discover that we can boost accuracy further by combining predictions across input presentation at multiple resolutions. Note that

CHIL '20, June 03–05, 2018, Toronto, ON

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

this approach requires running the multi-resolution pipeline at test time as well.

We explore two different approaches for *deterministic* temporal coarsening of irregular time series. First, inspired by interpolation networks [15, 45], we shorten the time series to a time series with regularly-spaced timestamps. Alternatively, we also propose the *cluster&count* operator to create a shorter version of the irregular time series by clustering the timestamps. The values of the new series' data points are the averages of the data points belonging to the corresponding clusters. We also count the number of data points in each cluster and append it to the new time series as an extra feature.

We can use the two proposed coarsening operators to create the MultiResolution network with Shared Weights (MRSW) model. We select four instances of time series using four different coarsening probabilities. We process all versions with the same neural network and use the attention mechanism to average their predictions. The proposed model not only benefits from the event-clustering invariance and is more robust to overfitting. It shares weights across multiple resolutions of time series [8, 42] without a significant increase in the number of parameters.

Our experiments on two benchmark clinical prediction tasks [20] defined on the publicly available MIMIC-III dataset [25] highlight the usefulness of the proposed operators. We test the performance of a large convolutional neural network with and without data augmentation and show its success in improving the generalization of the model. Second, we show that data augmentation improves robustness of the models against adversarial perturbations. Finally, our experiments show that MRSW improves the mean average precision on the benchmark mortality prediction task from 51.53% to 53.92%.

Our main contributions are as follows:

- For the first time, we identify the temporal-clustering invariance in healthcare time series.
- We propose a fast data augmentation technique to improve robustness of neural networks training by making them invariant to temporal-clustering.
- We propose a new network architecture MRSW that attends different temporal-clusterings of the input time series.
- Our extensive experiments show that MRSW achieves the state of the art accuracy on the two key tasks of the [20] benchmark.

2 METHODOLOGY

In this section, we describe the structure of sequential EHR data, introduce required notation, and then describe our proposed methods.

EHR Structure and our Notation. The EHR data for each patient consists of a time-labeled multivariate sequence observations. Assuming that we use r different variables, the time series data for the n-th patient $X^{(n)}$ (out of N total patients) can be represented by a sequence of $T^{(n)}$ tuples $(t_i^{(n)}, \mathbf{x}_i^{(n)}) \in \mathbb{R} \times \mathbb{R}^r, i = 1, \ldots, T^{(n)}$. The timestamp $t_i^{(n)}$ denotes the time of the i-th visit of the n-th patient, $\mathbf{x}_i^{(n)}$ denotes an r-dimensional feature vector that typically takes the value of the observed variable(s) at the corresponding component(s) (the treatment of missing variables is addressed in greater detail below), and $T^{(n)}$ denotes the number of visits of the n-th patient. To minimize clutter, we drop the superscript (n) whenever it is unambiguous. Depending on which precise task we are addressing, the objective of our predictive models is to predict either (i) the label corresponding to each time step $\mathbf{y}_i \in \{0, 1\}^s$ given only the left-ward context, or (ii) to predict a sequence-level label at the end of the sequence $\mathbf{y} \in \{0, 1\}^s$. Commonly, as in the learning to diagnose task, the prediction task is multilabel, i.e. $s > 1$.

Algorithm 1 Fast Data Augmentation

Input: patient sequence $\{(t_i, \mathbf{x}_i, c_i)\}_{i=1}^{T}$,
largest clustering probability p_{high}, Weighted or unweighted augmentation
Output: transformed sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$.
if Weighted **then**
  $\boldsymbol{\Delta} \leftarrow [t_2 - t_1, \ldots, t_T - t_{T-1}]$.
  $\widetilde{\boldsymbol{p}}_\Delta \leftarrow [1/\Delta_1, \ldots, 1/\Delta_{T-1}]$.
  $\boldsymbol{p}_\Delta = \widetilde{\boldsymbol{p}}_\Delta / (\mathbf{1}^\top \widetilde{\boldsymbol{p}}_\Delta)$.
else
  $\boldsymbol{p}_\Delta \leftarrow \mathbf{1}_{T-1}/(T-1)$.
end if
Draw $p \sim \text{Unif}(0, p_{\text{high}})$
$E \leftarrow \lceil p \cdot T \rceil$ samples without replacement from Multinomial($\boldsymbol{p}_\Delta$)
Create clusters $C_{i'}$ for $i' = 1, \ldots, T'$ by assigning every pair $(i, i+1)$ to the same cluster if $i \in E$.
for $i' = 1$ **to** T' **do**
  $t'_i \leftarrow \text{mean}(\{t : t \in C_{i'}\})$
  $\mathbf{x}'_i \leftarrow \text{mean}(\{\mathbf{x} : \mathbf{x} \in C_{i'}\})$
  $c'_i \leftarrow \text{sum}(\{c : c \in C_{i'}\})$
end for
Return: sorted sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$ based on t'.

Learning Tasks. Our proposed methods can be used in both learning to diagnose (L2D) [33] and encounter sequence modeling (ESM) [10]. In the L2D task, the input vector $\mathbf{x}_i$ consists of continuous clinical measurements. If there are r different measurements, then $\mathbf{x}_i \in \mathbb{R}^r$. The goal of L2D is, given an input sequence $\mathbf{x}_1, \ldots, \mathbf{x}_T$, to predict the occurrence of a specific disease ($s = 1$) or multiple diseases ($s > 1$). Without loss of generality, we will describe the algorithm for L2D, as ESM can be seen as a special case of L2D where we make a prediction at each timestamp t given the patient's history before t.

Invariant Learning. Suppose $\mathcal{T}(X)$ denotes a transformation of the patient data X. If $P(y|\mathcal{T}(X)) = P(y|X)$, prediction of y with X is invariant to transformation $\mathcal{T}$ [2]. If the relationship holds approximately, i.e. $P(y|\mathcal{T}(X)) \approx P(y|X)$, we have a more relaxed approximate invariance. Given powerful estimators such as RNNs, we can augment the training data using the transformation and achieve a $\mathcal{T}-$invariant model. In the rest of this section, we introduce the temporal-clustering approximate invariance, use it in the augmentation setting, and propose an multiresolution model based on it.

2.1 Fast Data Augmentation

Data augmentation is the process of creating distorted instances of data points, nominally increasing the size of the dataset. When applied to exploit known invariances, this technique is well-known to reduce overfitting, and has become a standard part of the applied computer vision [31], and in automatic speech recognition, where the technique is dubbed *multicondition training* [3]. We design a data augmentation procedure for healthcare time series demonstrating that temporal clustering is indeed an invariance worth exploiting in real-world healthcare time series.

To formally describe the augmentation operator, we append a count variable to each tuple in the data, i.e. $(t_i, \mathbf{x}_i) \rightarrow (t_i, \mathbf{x}_i, c_i)$. Clearly, $c_i = 1$ for $i = 1, \ldots, T$ for all original sequences. Let $\mathbb{S}_T$ denote the space of all patient sequences of length T and a count-augmented patient sequence, which is a member of $\mathbb{S}_T$, as X. The data augmentation operation is a probabilistic mapping $\mathrm{da}_p(\cdot) : \mathbb{S}_T \mapsto \mathbb{S}_{T'}$, where $T' = T - \lceil p \cdot T \rceil$ and $\lceil \cdot \rceil$ is the ceiling operator. We draw the coarsening probability p randomly according to $p \sim \mathrm{Unif}(0, p_{\text{high}})$, The upper bound on the transformed sequence length $(1 - p_{\text{high}})T$ corresponds to the degree of regularization conferred by this approach; lower values of p_{high} give greater regularization.

Given a clustering probability p, for a sequence of length T, we choose $\lceil p \cdot T \rceil$ intervals at random without replacement and combine the events at both ends of the selected intervals. The details of the algorithm are provided in Algorithm 1. We further explore a *weighted* version of the data augmentation by joining the closer events likelier. For the weighted data augmentation, we create a time interval vector of length $T - 1$: $\mathbf{\Delta} = [t_2 - t_1, \ldots, t_T - t_{T-1}]$. We create a probability value for $T - 1$ intervals by defining $\widetilde{\boldsymbol{p}}_\Delta = [1/\Delta_1, \ldots, 1/\Delta_{T-1}]$ and normalizing it as $\boldsymbol{p}_\Delta = \widetilde{\boldsymbol{p}}_\Delta / (\mathbf{1}^\top \widetilde{\boldsymbol{p}}_\Delta)$. We define a multinominal distribution using the probability vector $\boldsymbol{p}_\Delta$ and draw $\lceil p \cdot T \rceil$ samples from it without replacement. We create the final clustering by assigning the data points on the ends of each drawn interval to the same cluster.

Our data augmentation is closely related and inspired by the idea of fractional pooling [18]. We can model irregular time series as graphs whose edge weights are defined by the time interval between two events. Processing such graphs with convolutional graph neural networks [6, 21, 39], the graph pooling operator will correspond to the clustering operation. If we choose fractional pooling, we will have similar stochastic coarsening operations that we have used here for data augmentation.

The data augmentation combats overfitting by providing distorted versions of the data to the training algorithm. In typical data augmentation schemes the model only uses the unperturbed original data at test time. In our augmentation scheme experiments confirm benefits to providing access to the coarser-grained versions of the data at test time. To begin, we define our new deterministic time series coarsening operators.

2.2 The Coarsening Operations

In designing the coarsening operator, we seek transformations from given data sequences to various different but plausible sequence. We exploit our intuition that multiple data points might be collected during a single visit or across multiple visits. This leads us to the idea of creating different resolutions of the sequence by clustering the timestamps into a number of groups, (resulting in a shorter sequence than the original). We note that clustering might significantly change the point process properties of the sequence. Thus, we append a *count* variable to each cluster that indicates the number of data points assigned to that cluster. We consider two possibilities for coarsening of irregular time series: *regular-grid-based* and *clustering-based.*

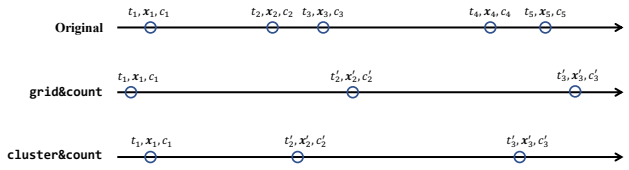


Figure 1: Visualization of the coarsening operators. $grid\&count_{0.6}(\cdot)$ clusters data points into uniformly spaced data points over time. $\boldsymbol{cluster\&count}_{0.6}(\cdot)$ produces another irregular time series with the new time stamps $t'_2 = (t_2 + t_3)/2$ and $t'_3 = (t_4 + t_5)/2$. In both cases we have $\mathbf{x}'_2 = (\mathbf{x}_2 + \mathbf{x}_3)/2$ and $\mathbf{x}'_3 = (\mathbf{x}_4 + \mathbf{x}_5)/2$. The count variables take values $c'_1 = 1$ and $c'_2 = c'_3 = 2$.

Algorithm 2 Grid&count

Input: patient sequence $\{(t_i, \mathbf{x}_i, c_i)\}_{i=1}^T$,
clustering probability p, and observation interval $[t_L, t_R]$.
Output: transformed sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$.
$T' \leftarrow \lceil p \cdot T \rceil$.
$t_{i'+1} \leftarrow t_L + i \cdot (t_R - t_L)/(T' - 1)$ for $i' = 0, \ldots, T' - 1$.
Initialize clusters $C_{i'} = \emptyset$ for $i' = 1, \ldots, T'$.
for $i = 1$ **to** T' **do**
  $i^\star \leftarrow \mathrm{argmin}_{i'} \{|t_i - t'_{i'}| \text{ for } i' = 1, \ldots, T'\}$.
  $C_{i^\star} \leftarrow C_{i^\star} \cup \{(t_i, \mathbf{x}_i, c_i)\}$
end for
for $i' = 1$ **to** T' **do**
  $t'_i \leftarrow \mathrm{mean}(\{t : t \in C_{i'}\})$
  $\mathbf{x}'_i \leftarrow \mathrm{mean}(\{\mathbf{x} : \mathbf{x} \in C_{i'}\})$
  $c'_i \leftarrow \mathrm{sum}(\{c : c \in C_{i'}\})$
end for
Return: sorted sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$ based on t'.

Algorithm 3 Cluster&count

Input: patient sequence $\{(t_i, \mathbf{x}_i, c_i)\}_{i=1}^T$,
clustering probability p.
Output: transformed sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$.
$T' \leftarrow \lceil p \cdot T \rceil$.
Create clusters $C_{i'}$ for $i' = 1, \ldots, T'$ by clustering $\{t_i\}_{i=1}^T$ into T' clusters using k-means.
for $i' = 1$ **to** T' **do**
  $t'_i \leftarrow \mathrm{mean}(\{t : t \in C_{i'}\})$
  $\mathbf{x}'_i \leftarrow \mathrm{mean}(\{\mathbf{x} : \mathbf{x} \in C_{i'}\})$
  $c'_i \leftarrow \mathrm{sum}(\{c : c \in C_{i'}\})$
end for
Return: sorted sequence $\{(t'_i, \mathbf{x}'_i, c'_i)\}_{i=1}^{T'}$ based on t'.

With the grid-based coarsening, our aim is to convert an irregular time series to a shorter time series with regularly spaced timestamps. We first select a set of timestamps, and then assign each event to the closest timestamp. Next, we compute the average values among events assigned to each new timestamp to calculate the transformed time series. If no event is assigned to a particular timestamp, we leave the values as all zeros. Finally, we append the cluster size as an extra feature to the feature vector. Notice that grid-based coarsening, similar to interpolation networks [15, 45], converts irregular time series to regular ones, which can make it easier for analysis in the next steps. The details of the *grid&count* operation are provided in Algorithm 2.

Our proposed alternative is a clustering-based coarsening operator, which we denote *cluster&count*. Here, we execute a temporal clustering of the sequence of events in the original sequence, yielding taking the cluster centers as new time stamps. The count variables $c_{i'}$ in the transformed sequence represent the number of points in the i'th cluster. Assuming that all categorical features have been encoded with one-hot or similar encoding, the new feature vectors is the average of all vectors in the same cluster.

Algorithms 2 and 3 show the details of the *grid&count* and *cluster&count* operations, respectively. Figure 1 visualizes the operators in a simple case. In both coarsening operations, we ignore the missing data during aggregation (inside the last for loop in the algorithms). If a variable is missed in all events belonging to a cluster, we impute it with zero, following the observations in [34].

2.3 A Multi-resolution Network

In this section, we propose to generate multiple resolutions of the input data using the coarsening operators described in the previous section for use by our predictive models. Our model exploits multi-resolution via an attention mechanism. To avoid an unnecessary explosion in the number of parameters, we share weights of our model across all resolutions.

Formally, let $\boldsymbol{f_\theta} : \mathbb{S}_T \mapsto [0,1]^s$ denote a classifier parameterized by $\boldsymbol{\theta}$, which maps a patient sequence to probabilities of labels being 1. Given K different clustering factors p_k for $k = 1, \ldots, K$, we define the MultiResolution network with Shared Weights (MRSW) as follows:

$$\boldsymbol{g_{\theta,\beta}}(X) = \sum_{k=1}^{K} \alpha_{k,\boldsymbol{\beta}}(X) \boldsymbol{f_\theta}(C_{p_k}(X)), \tag{1}$$

where $\alpha_{k,\boldsymbol{\beta}}(\cdot)$ for $k = 1, \ldots, K$ are the attention values. The coarsening operator $C_{p_k}(X)$ can be either of grid-based or clustering based operators; we denote the corresponding networks by *MRSW-Grid* and *MRSW-Cluster*, respectively. The attention generation parameters $\boldsymbol{\beta}$ are the only additional parameters of this classifier. We can either use the intermediate hidden representations obtained by $\boldsymbol{f_\theta}$ from the one of $C_{p_k}(X)$ or use simple features such as length of time series. Given the pre-determined cluster lengths, we can precompute the $C_{p_k}(X)$ for all sequences and avoid the clustering cost at the run time. In our experiments, we use MRSW with $K = 4$ and $p_1 = 1, p_2 = 1/2, p_3 = 1/4$, and $p_4 = 1/8$. The exponentially decreasing p_k is intended to provide access to a wide range of resolutions of the input.

Unlike Razavian and Sontag [42], who use a different CNNs for each resolution of the data, MRSW shares the weights across all resolutions of the data and reduces the number of model parameters. We can view MRSW as a mixture model [24] and use the latent-variable inference tools to estimate the class probabilities instead of attention variables. In this work, we prefer to view the mixture weights as attention weights for computational simplicity. MRSW may potentially be more robust to jitters in the timestamps of the sequences [40].

We note a connection between the proposed MRSW and existing work that studies symmetries and invariances in neural networks [12, 16, 29]. Similar to the group convolutions, we compute the output of the neural network on the input data under different transformations. In contrast to the group convolutions, we apply the transformation *grid&count* or *cluster&count* to the data rather than the convolutional filters. Note that the neither of the coarsening operations define a group because it is not invertible due to information loss during the clustering operation. However, the counting operation attempts to preserve more information—related to the point process—during the coarsening process.

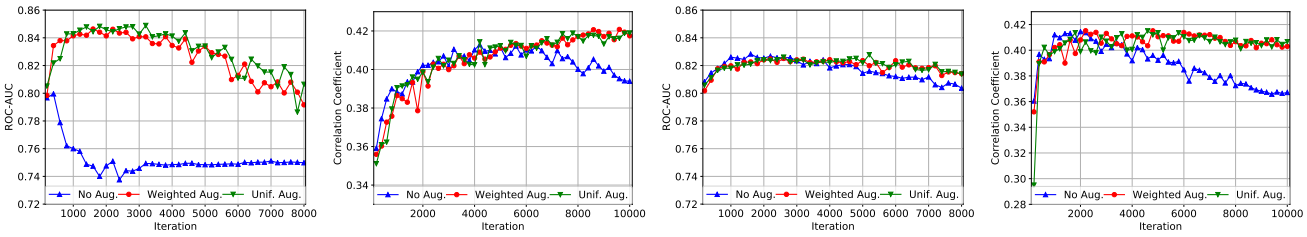
Base learners. We use the RNN base learners as $\boldsymbol{f_\theta}$ in Eq. (1) as they are the common deep neural networks for predictive modeling in healthcare time series [10, 33]. In this architecture, after linear embedding, we opt to use two layers of Gated Recurrent Units (GRU) [9] (instead of the better-known Long Short-Term Memory (LSTM) cells [22]) owing to their comparative parsimony with respect to parameters. For the prediction layer, we use a residual block with fully connected units. We provde further details of the architectures in Appendix A.

3 RELATED WORKS

Time deformation. One well-studied invariance in irregular time series is robustness to the deformations in the timing of the events. Dynamic Time Warping [26, 44, 52] is a popular traditional method to incorporate invariance to the exact timing of events. In more recent work, Oh et al. [40] propose to learn new timestamps for the events to make neural networks invariant to jitters in event times. Given the similarity between the coarsening operators and the one-dimensional pooling operation, we expect the coarsening operation provide a degree of time deformation invariance.

Errors in measurements. Measurement errors can happen both due to the devices recording and storing measurements and due to human error in registering values. Adversarial training [17] is one effective way to make algorithms less sensitive to small errors in the input data. If we attach the true timing of the events as a feature to the event features, adversarial training can potentially make the algorithm robust to jitters in the timing of the events too similar to [40]. Given that the proposed coarsening operators aggregate the values in the clusters, we expect to achieve a degree of invariance to small changes in measurements too.

Missing data. Often, not all variables are measured for all patients at all times [7, 34]. For data missing completely at random, we might seek to make our models robust via dropout training [48], randomly zeroing out a subset of variables in the input time series



(a) Mortality, CNN (b) Length of stay, CNN (c) Mortality, RNN (d) Length of stay, RNN

Figure 2: Augmenting the training data via fast temporal-clustering not only prevents overfitting in large CNN model but also does not significantly hurt the small RNN model. We demonstrate the behavior of the proposed data augmentation on two tasks: mortality and length of stay (LoS) prediction. The CNN model is intentionally chosen to be overparameterized. We choose an overparameterized CNN with and without *cluster&count* augmentation ($p_{\text{high}} = 0.5$) and SGD with fixed learning rate and momentum of 10^{-2} and 0.95, respectively. To show smoother curves, we show the average of eight runs with random initializations for each case.

and augment the data using the newly created data points. Other approaches include inputation via graphical models, or incorporating missing value indicators [7, 34].

Censoring. Medical data is often censored, i.e., we do not observe the full history for all patients [28]. For example, in the right censoring scenario, if a patient stays alive at the time of a survival analysis task, her future will not be available to the algorithm. Thus, we will not have the full-length records of all patients in our dataset. Target replication [13, 33, 37] is a technique that requires the algorithm to predict the final outcome at any point in the history of a patient. Thus, it aims at making the learning algorithm robust to right censoring.

4 EXPERIMENTS

We evaluate the proposed algorithms on two L2D benchmark tasks: in-hospital mortality and length of stay (LoS) prediction [20], which are defined on the publicly available MIMIC-III dataset [25]. Both predictive tasks are widely used in estimating patients' clinical risk and managing costs in hospitals. For example, the LACE clinical risk scores [5, 19, 50, 51] require an estimate of the length of stay of a patient soon after they are admitted to the hospital. Details of feature extraction and cohort construction can be found in the following benchmark paper [20] and we provide a brief summary below.

4.1 Dataset, normalization, and training details

Dataset. We follow Harutyunyan et al. [20], extracting the patient sequences from the MIMIC-III database and partition the data into training and testing sets. The benchmark data for mortality and LoS prediction tasks have 17,902/3,236 and 35,344/6,225 training/test data points, respectively. We do not use the benchmark discretization and normalization steps and normalize the irregular time series by ourselves as follows. In the length of stay prediction task, we do not use the dataset expansion technique used by the benchmark; i.e., for both training and testing we use 24 hours of patient history.

Normalization of real values. Noting the existence of outliers in the data, we compute the robust mean and variance by excluding the values below the 2nd percentile and above 98th percentile. Using the estimated robust mean and variance, we normalize the real-valued quantities to zero mean and unit variance. Additionally, for robustness, we Winsorize the real values by thresholding them within the limits of 2nd and 98th percentiles.

Ordinal encoding. We use the unary coding [14, 36] for the ordinal quantities such as the Glasgow Coma Score instead of the one-hot encoding in [20]. A k-level ordinal variable with real values $a_1 < a_2 < \ldots < a_k$ is encoded using a binary vector $\boldsymbol{b}$ of size $k-1$. We represent each value a_ℓ for $\ell = 1, \ldots, k$ by setting the first $\ell - 1$ elements of $\boldsymbol{b}$ to 1. This approach encodes $a_{\ell+1} - a_\ell$ values and is able to model large real values for a_k. We also quantize the count variables c in bins of $(0-1]$, $(1-2]$, $(2,4]$, $(4,\infty)$ and encode them using the unary coding. After appending the encoded time and count vectors, the resulting feature vectors are 51-dimensional. We use zero-filling to handling missing variables because it is simple and efficient [34].

Training details. We hold out 15% of the training data as a validation set for tuning the hidden layer sizes and hyperparameters and report the test results based on the best validation performance. For optimization, we use Adam [27] with the AMSGrad modification [43] with batch size of 100. We halve the learning rate after plateauing for 10 epochs (determined on validation data) and stop training after the learning rate drops below 5×10^{-6}. We run each algorithm 8 times with different random initializations and report the result of the run with the highest validation accuracy. All of the proposed models are implemented in PyTorch [41].

Baselines. We compare our main accuracy results in Table 1 to the single-task results published in [20, 47]. Moreover, we design two new baselines to evaluate our preprocessing and data augmentation techniques. We report our baseline GRU network on the data preprocessed by the benchmark [20], denoted by "GRU (baseline preprocessing)". "GRU (our preprocessing)" shows the accuracy of our GRU model on the data with our special preprocessing describe earlier in this section. To compare the efficiency of our data augmentation to the existing data augmentation techniques discussed in Section 3. We train our GRU with a data augmentation policy that randomly misses variables with probability p_{miss} and randomly

left-cencors p_{censor} fraction of the time series. We tune these two probabilities using the random search.

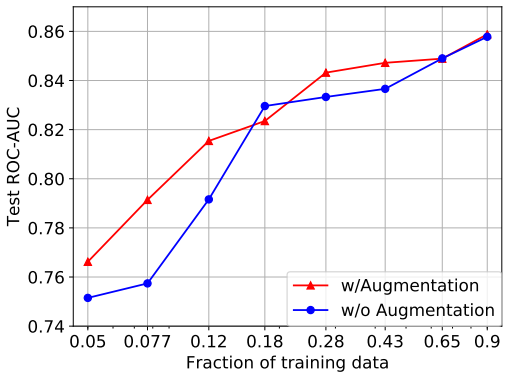
4.2 Results and analysis

Data augmentation. First, we test the impact of data augmentation in improving the robustness of training. An ideal data augmentation procedure needs to make complex models robust to overfitting, without hurting the performance of others. To show these desirable attributes for our proposed data augmentation, To show the benefits of our methods vis-a-vis overfitting, we show results on a large CNN that tends generally to overfit this data quickly. We use the SGD with fixed learning rate and momentum of 10^{-2} and 0.95, respectively. Results of this experiments are shown in Figure 2. To show smoother curves, we show the average of eight runs with random initializations for each case. Figures 2(a) and 2(b) show that data augmentation allows longer training time before the validation accuracy decreases. We also experiment with a GRU model less prone to overfitting. Figures 2(c) and 2(d) show that data augmentation helps the GRU model as well but to a lesser extent. More details on both network architectures are given in Appendix A.

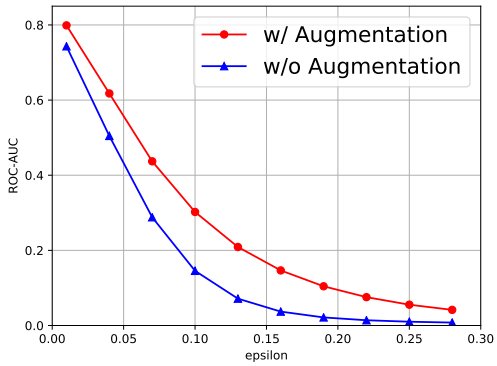
We experiment with both weighted and unweighted coarsening augmentation. The results Figure 2 indicated that weighted and unweighted augmentations have similar effects on the performance of the algorithms. Looking at the beginning of Figure 2(a), it is clear that the weighted augmentation provides a less diverse set of samples that are more similar to the actual data. It is a milder data augmentation, thus the performance improves faster early during the training. Given the milder nature of the weighted augmentation, it is a better candidate for use alongside other data augmentation techniques.

In Figure 3(a), we examine the impact of data augmentation as we increase the fraction of training data used during the training. Ignoring the seemingly anomalous data point in 0.18, the general trend indicates that initially when we are given only 5% of data, our data augmentation improves the test AUC by about 1 percentage point. The amount of improvement increases as we use a larger fraction of data when we use 0.077 and 0.12 fraction of training data points. As we further increase the fraction of training data, the gain by data augmentation shrinks. Note that this this result also indicates that the fact that in Figures 2(c) and 2(d) the gap between non-augmented and augmented training is negligible is only because of the size of the training data and the network in the tasks.

Our proposed temporal clustering-based data augmentation is more robust to the perturbations on the input features, as shown in Figure 3(b). To demonstrate the (relative) adversarial robustness, we consider its performance under attacks via the fast gradient sign method (FGSM) due to Goodfellow et al. [17], because it is simple and parameterized only with a single parameter ε. In this purturbation, we add the adversarial noise $\widetilde{\mathbf{x}} \leftarrow \mathbf{x} + \varepsilon \cdot \text{sign}\left(\frac{\partial \mathcal{L}(y, f(\mathbf{x}))}{\partial \mathbf{x}}\right)$ to the input, where $f(\cdot)$ and $\mathcal{L}(\cdot,\cdot)$ denote the prediction and binary cross-entropy loss functions, respectively. Then, we evaluate the accuracy of the classifier on the perturbed data $f(\widetilde{\mathbf{x}})$. We perform this test on the convolutional neural networks in the previous example on the mortality task. The results (Figure 3(b)) show that



(a) Learning Curve



(b) Input Sensitivity Analysis

Figure 3: *(a)* The learning curve with and without data augmentation on the mortality prediction task. The network is the same GRU network used in Figure 2. The x-axis in on a logarithmic scale. *(b)* Data augmentation using *cluster&count* improves robustness to adversarial examples. Here we show the ROC-AUC of the CNN network on the mortality task against FGSM attacks of varying perturbation strength.

the data augmentation via *cluster&count* does indeed increase the robustness to adversarial perturbations. The reason behind the robustness is that the time series transformed by *cluster&count* not only have new timestamps but also they have new values for the variables. Note that our analysis is not intended to show that our data augmentation creates robustness to adversarial attacks. Instead we offer these experiments simply to provide input sensitivity analysis.

Accuracy evaluation. We present the main accuracy results in Table 1. To quantify the randomness in the test set, we report the bootstrap (1000 runs) estimate of the mean and standard error of the evaluation measures. We measure accuracy of classification with two metrics: the area under receiver true-positive vs false-positive curve known as "ROC-AUC" and the area under precision recall curve known as mean average precision or "mAP". Because the

Table 1: Prediction accuracy results: bold numbers show the best results. Our results are reported in the form of 'mean (standard error)'. ⋆-marked experiments use a special dataset expansion technique.

Model	In-hospital Mortality Prediction		Length of Stay Prediction		
	ROC-AUC	mAP	Correlation	MAE	RMSE
LSTM [20]	0.8623	0.5153	–	94.00⋆	205.34⋆
SAnD [47]	0.857	0.518	–	–	200.93⋆
GRU (baseline preprocessing)	0.8556 (0.0003)	0.4950 (0.0009)	0.4093 (0.0005)	56.3557 (0.0459)	122.9606 (0.1683)
GRU (our preprocessing)	0.8642 (0.0003)	0.5263 (0.0009)	0.4199 (0.0005)	55.9366 (0.0504)	123.1219 (0.1760)
GRU-Augmented (ours)	0.8566 (0.0003)	0.5240 (0.0009)	0.4289 (0.0006)	55.9823 (0.0514)	123.1832 (0.1850)
GRU-Augmented (baseline)	0.8527 (0.0003)	0.5163 (0.0009)	0.4063 (0.0005)	56.4766 (0.0498)	125.1533 (0.1865)
MRSW-GRU-Grid	**0.8670 (0.0003)**	**0.5392 (0.0009)**	0.4328 (0.0005)	56.3363 (0.0501)	123.2220 (0.1627)
MRSW-GRU-Cluster	0.8665 (0.0003)	**0.5392 (0.0008)**	**0.4402 (0.0005)**	**55.5906 (0.0527)**	123.6223 (0.1774)

length of stay prediction is a regression task, we use three commonly used metrics: mean absolute error (MAE) and root mean squared error (RMSE) as un-normalized and linear correlation coefficient as a normalized accuracy measures.

As we can see in Table 1, (1) we confirm that our GRU network on our preprocessed data works better than the baselines [20, 47] because of our preprocessing techniques. (2) Our 'temporal-clustering'-based data augmentation significantly outperforms the baseline data augmentation. This implies that temporal-clustering generates augmented examples that are more similar to the actual data. (3) Finally, both variants of the proposed MRSW together with our preprocessing outperform the single-task baselines reported in [20, 47].

Two observations in the LoS prediction results in Table 1 stand out: (1) Our MAE and RMSE results in the LoS task is much better than the results reported in [20, 47]. Inspecting the benchmark preprocessing code, we realized that the authors have expanded the training dataset by creating examples of patients with time series longer than 24 hours and LoS value recalculated at the end of training period. We observed that this dataset expansion creates significant bias in the results and did not use it; (2) Given the large discrepancy between RMSE and MAE results, the RMSE values are likely to be impacted by the tail values. The standard errors also indicate that differences in the results are unlikely to be significant.

In terms of speed, MRSW is not significantly slower than the corresponding weak learner. We run the coarsening operators on the data and persist the clusters before the training of MRSW and only compute the batches only fly during the training. In terms of training convergence speed, Figure 4 shows that MRSW converges faster compared to the weak learner that it uses.

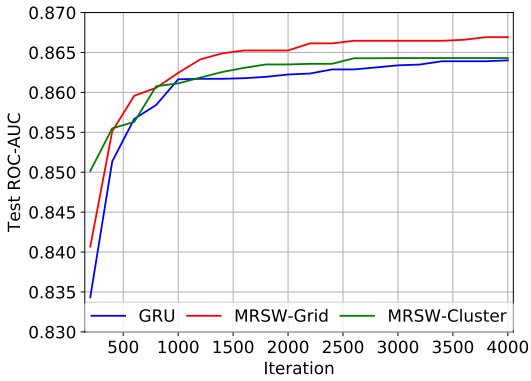


Figure 4: The MRSW-Grid model with the GRU weak-learner converges faster per iteration on the mortality prediction task. This plot shows the average best test accuracy as we train for more iterations. Note that in Table 1 we have used the validation accuracy for selection of the best model, thus the results are different.

5 CONCLUSIONS AND FUTURE WORK

In this work, we proposed to exploit a postulated temporal clustering invariance and examined the benefits of the derived methods for making predictions given time series of clinical healthcare data. Our proposed data augmentation techniques exploit this invariance to prevent overfitting in neural networks. Moreover, we demonstrate that a multi-resolution network can improve predicted accuracy by acting simultaneously upon multiple resolutions of data. Our methods in this paper represent clusters by averaging representations. In future work, we might learn the aggregation function too. Additionally, we might seek to understand the relationship between temporal clustering invariance and other properties of multivariate clinical time series data.

REFERENCES

[1] Ahmed M Alaa, Scott Hu, and Mihaela van der Schaar. 2017. Learning from clinical judgments: Semi-markov-modulated marked hawkes processes for risk prognosis. *arXiv:1705.05267* (2017).
[2] Mohammad Taha Bahadori and Layne C Price. 2019. Discovering Invariances in Healthcare Neural Networks. *arXiv preprint arXiv:1911.03295* (2019).
[3] Jon Barker, Martin Cooke, and Phil Green. 2001. Robust ASR based on clean speech models: An evaluation of missing data techniques for connected digit recognition in noise. In *EUROSPEECH*.
[4] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. 2017. Patient subtyping via time-aware LSTM networks. In *KDD*. ACM, 65–74.
[5] Eli Ben-Chetrit, Chen Chen-Shuali, Eran Zimran, Gabriel Munter, and Gideon Nesher. 2012. A simplified scoring tool for prediction of readmission in elderly patients hospitalized in internal medicine departments. *Isr Med Assoc J* 14, 12

(2012).
[6] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2014. Spectral networks and locally connected networks on graphs. In *ICLR*.
[7] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. 2018. Recurrent neural networks for multivariate time series with missing values. *Scientific reports* (2018).
[8] Zhengping Che, Sanjay Purushotham, Guangyu Li, Bo Jiang, and Yan Liu. 2018. Hierarchical Deep Generative Models for Multi-Rate Multivariate Time Series. In *ICML*. 783–792.
[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*.
[10] Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F Stewart, and Jimeng Sun. 2016. Doctor ai: Predicting clinical events via recurrent neural networks. In *MLHC*. 301–318.
[11] Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter Stewart. 2016. Retain: An interpretable predictive model for healthcare using reverse time attention mechanism. In *NIPS*.
[12] Taco Cohen and Max Welling. 2016. Group equivariant convolutional networks. In *ICML*.
[13] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *NIPS*. 3079–3087.
[14] Ila R Fiete and H Sebastian Seung. 2007. Neural network models of birdsong production, learning, and coding. In *New Encyclopedia of Neuroscience*.
[15] Joseph Futoma, Sanjay Hariharan, Katherine Heller, Mark Sendak, Nathan Brajer, Meredith Clement, Armando Bedoya, and Cara O'Brien. 2017. An Improved Multi-Output Gaussian Process RNN with Real-Time Validation for Early Sepsis Detection. In *MLHC*.
[16] Robert Gens and Pedro M Domingos. 2014. Deep symmetry networks. In *NIPS*. 2537–2545.
[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv:1412.6572* (2014).
[18] Benjamin Graham. 2014. Fractional max-pooling. *arXiv preprint arXiv:1412.6071* (2014).
[19] Andrea Gruneir, Irfan A Dhalla, Carl van Walraven, Hadas D Fischer, Ximena Camacho, Paula A Rochon, and Geoffrey M Anderson. 2011. Unplanned readmissions after hospital discharge among patients identified as being at high risk for readmission using a validated predictive algorithm. *Open Medicine* 5, 2 (2011), e104.
[20] Hrayr Harutyunyan, Hrant Khachatrian, David C Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific Data* 6, 1 (2019), 96.
[21] Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data. *arXiv:1506.05163* (2015).
[22] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* (1997).
[23] Kazi T Islam, Christian R Shelton, Juan I Casse, and Randall Wetzel. 2017. Marked Point Process for Severity of Illness Assessment. In *MLHC*. 255–270.
[24] Yangfeng Ji, Gholamreza Haffari, and Jacob Eisenstein. 2016. A Latent Variable Recurrent Neural Network for Discourse Relation Language Models. In *NAACL-HLT*. 332–342.
[25] Alistair EW Johnson, Tom J Pollard, Lu Shen, H Lehman Li-wei, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G Mark. 2016. MIMIC-III, a freely accessible critical care database. *Scientific data* 3 (2016), 160035.
[26] Eamonn Keogh and Chotirat Ann Ratanamahatana. 2005. Exact indexing of dynamic time warping. *KAIS* (2005).
[27] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv:1412.6980* (2014).
[28] John P Klein and Melvin L Moeschberger. 2006. *Survival analysis: techniques for censored and truncated data*. Springer Science & Business Media.
[29] Risi Kondor and Shubhendu Trivedi. 2018. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *arXiv:1802.03690* (2018).
[30] Thijs Kooi and Nico Karssemeijer. 2017. Classifying symmetrical differences and temporal change for the detection of malignant masses in mammography using deep neural networks. *Journal of Medical Imaging* 4, 4 (2017), 044501.
[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.
[32] Steven Cheng-Xian Li and Benjamin M Marlin. 2015. Classification of Sparse and Irregularly Sampled Time Series with Mixtures of Expected Gaussian Kernels and Random Features.. In *UAI*.
[33] Zachary C Lipton, David C Kale, Charles Elkan, and Randall Wetzel. 2016. Learning to diagnose with LSTM recurrent neural networks. In *ICLR*.
[34] Zachary C Lipton, David C Kale, and Randall Wetzel. 2016. Modeling missing data in clinical time series with RNNs. *MLHC* (2016).
[35] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. 2017. Dipole: Diagnosis prediction in healthcare via attention-based bidirectional recurrent neural networks. In *KDD*.
[36] Jordan M Moore, Tamás Székely, József Büki, and Timothy J DeVoogd. 2011. Motor pathway convergence predicts syllable repertoire size in oscine birds. *PNAS* 108, 39 (2011), 16440–16445.
[37] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *CVPR*.
[38] Phuoc Nguyen, Truyen Tran, Nilmini Wickramasinghe, and Svetha Venkatesh. 2017. Deepr: A Convolutional Net for Medical Records. *BHI* (2017).
[39] Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. 2016. Learning convolutional neural networks for graphs. In *ICML*. 2014–2023.
[40] Jeeheh Oh, Jiaxuan Wang, and Jenna Wiens. 2018. Learning to Exploit Invariances in Clinical Time-Series Data using Sequence Transformer Networks. In *MLHC*.
[41] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.
[42] Narges Razavian and David Sontag. 2015. Temporal convolutional neural networks for diagnosis from lab tests. *arXiv:1511.07938* (2015).
[43] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. 2018. On the convergence of adam and beyond. In *ICLR*.
[44] Hiroaki Sakoe and Seibi Chiba. 1978. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Trans. Acoust., Speech, Signal Process* 26, 1 (1978), 43–49.
[45] Satya Narayan Shukla and Benjamin M. Marlin. 2019. Interpolation-Prediction Networks for Irregularly Sampled Time Series. In *ICLR*.
[46] Hossein Soleimani, Adarsh Subbaswamy, and Suchi Saria. 2017. Treatment-response models for counterfactual reasoning with continuous-time, continuous-valued interventions. In *UAI*.
[47] Huan Song, Deepta Rajan, Jayaraman J Thiagarajan, and Andreas Spanias. 2018. Attend and Diagnose: Clinical Time Series Analysis using Attention Models. In *AAAI*.
[48] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *JMLR* 15, 1 (2014), 1929–1958.
[49] Truyen Tran, Wei Luo, Dinh Phung, Sunil Gupta, Santu Rana, Richard Lee Kennedy, Ann Larkins, and Svetha Venkatesh. 2014. A framework for feature extraction from hospital medical data with applications in risk prediction. *BMC bioinformatics* 15, 1 (2014), 425.
[50] Carl van Walraven, Irfan A Dhalla, Chaim Bell, Edward Etchells, Ian G Stiell, Kelly Zarnke, Peter C Austin, and Alan J Forster. 2010. Derivation and validation of an index to predict early death or unplanned readmission after discharge from hospital to the community. *Can. Med. Assoc. J.* (2010).
[51] Carl van Walraven, Jenna Wong, and Alan J Forster. 2012. LACE+ index: extension of a validated index to predict early death or urgent readmission after hospital discharge using administrative data. *Open Medicine* 6, 3 (2012), e80.
[52] T.K. Vintsyuk. 1968. Speech discrimination by dynamic programming. *Cybernetics* 4, 1 (1968), 52–57.
[53] Shuai Xiao, Junchi Yan, Xiaokang Yang, Hongyuan Zha, and Stephen M Chu. 2017. Modeling the Intensity Function of Point Process Via Recurrent Neural Networks. In *AAAI*.
[54] Kaiping Zheng, Wei Wang, Jinyang Gao, Kee Yuan Ngiam, Beng Chin Ooi, and Wei Luen James Yip. 2017. Capturing Feature-Level Irregularity in Disease Progression Modeling. In *CIKM*.

A DETAILS OF THE BASE LEARNERS' ARCHITECTURE

GRU models. Our GRU model is a two layers GRU layers followed by a residual block for prediction. The residual block is in the form of relu(bnorm_2(fc_2(relu(bnorm_2(fc_2(**x**)))))) + fc_3(**x**). We perform light hyperparameter optimization on the size of hidden vector of GRU.

The CNN model. The CNN model first embeds the input into a higher dimensional vector. Next, it applies a sequence of convolutional residual blocks with progressively larger dilations to the embedded vector. Finally, we flatten the last hidden tensor and apply a batch normalization and dropout layer before fully connected prediction layer.

In a residual block with dilation factor of d, we apply d dilations in the pair of convolutions and $2d$ dilation in the skip connection. The kernel size for all 1d-convs is set to 3.

In the experiments for Figure 2, we intentionally large CNN model. Its specific settings are as follows:

(1) Embedding (dim=200)
(2) ResBlock (200, 180, 160, kernel_size=3, dil=1)
(3) ResBlock (160, 140, 120, kernel_size=3, dil=10)
(4) ResBlock (120, 100, 80, kernel_size=3, dil=25)
(5) FC(1280, 1)

On the length of stay task, we use dilation sequence of (1, 3, 3), because the sequences are shorter.