UNIVERSITY OF CALIFORNIA

Los Angeles

Deep Unsupervised & Semi-supervised Methods for Health Applications

A dissertation submitted in partial satisfaction

of the requirements for the degree

Doctor of Philosophy in Computer Science

by

Sajad Darabi

2021

ABSTRACT OF THE DISSERTATION

Deep Unsupervised & Semi-supervised Methods for Health Applications

by

Sajad Darabi
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2021
Professor Majid Sarrafzadeh, Chair

Recent success in machine learning for various applications such as image classification and language generation via deep learning has encouraged similar development in other domains. In particular with the ubiquity of health sensors, troves of data are being collected from which useful information can be extracted to improve overall quality of life. For example, electronic health records (EHR) have become widespread across hospitals where many data modalities are collected during patient care. Although these datasets are befitting to the supervised learning framework, often due to limited annotated data, extensive missingness, and the temporal nature of the data, supervised models often generalize poorly. We address this by introducing unsupervised and semi-supervised methods that leverage unlabeled raw patient data to help downstream task generalization. To demonstrate the effectiveness of our proposed methods, and show their utility on real-world public medical datasets, including cohorts from hospitals, intensive care units, and wards. The set of methods introduced are simple and effective at improving downstream performance.

The dissertation of Sajad Darabi is approved.

Wei Wang

Kai-Wei Chang

Sriram Sankararaman

Majid Sarrafzadeh, Committee Chair

University of California, Los Angeles

2021

To what ought to be and its actuality

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

VITA

2016-2021	Ph.D. Graduate Student, University of California, Los Angeles,
2016	Bachelor of Electrical Engineering, McGill University, Montreal, Quebec, Canada
2010-2012	Wood Working, Montreal, Quebec, Canada

PUBLICATIONS

Darabi, S., Kachuee, M., Fazeli, S., Sarrafzadeh, M. (2020). TAPER: Time-Aware Patient EHR Representation. IEEE Journal of Biomedical and Health Informatics.

Darabi, S., Kachuee, M., Sarrafzadeh, M. (2019). Unsupervised Representation for EHR Signals and Codes as Patient Status Vector. arXiv preprint arXiv:1910.01803.

Kachuee, M., **Darabi, S.**, Moatamed, B., Sarrafzadeh, M. (2018). Dynamic feature acquisition using denoising autoencoders. IEEE transactions on neural networks and learning systems, 30(8), 2252-2262.

Kachuee, M., Goldstein, O., Karkkainen, K., **Darabi, S.**, Sarrafzadeh, M. (2019). Opportunistic learning: Budgeted cost-sensitive learning from data streams. arXiv preprint arXiv:1901.00243.

Kachuee, M., Hosseini, A., Moatamed, B., **Darabi, S.**, Sarrafzadeh, M. (2017, November). Context-aware feature query to improve the prediction performance. In 2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP) (pp. 838-842). IEEE.

Kachuee, M., Karkkainen, K., Goldstein, O., **Darabi, S.**, Sarrafzadeh, M. (2020). Generative Imputation and Stochastic Prediction. IEEE Transactions on Pattern Analysis and Machine Intelligence.

Moatamed, B., **Darabi, S.**, Gwak, M., Kachuee, M., Metoyer, C., Linn, M., Sarrafzadeh, M. (2017, November). Sport analytics platform for athletic readiness assessment. In 2017 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT) (pp. 156-159). IEEE.

Darabi, S., Moatamed, B., Huang, W., Gwak, M., Metoyer, C., Linn, M., Sarrafzadeh, M. (2017, November). Heart rate compression time reduction method for HRV monitoring in athletes. In 2017 IEEE Healthcare Innovations and Point of Care Technologies (HI-POCT) (pp. 152-155). IEEE.

Moatamed, Babak, Sajad Darabi, and Majid Sarrafzadeh. "Scoring System for Conditioning and Wellness Assessment in Athletic Population." 2019 IEEE Healthcare Innovations and Point of Care Technologies,(HI-POCT). IEEE, 2019

Darabi, S., Belbahri, M., Courbariaux, M., Nia, V. P. (2018). Bnn+: Improved binary network training. arXiv preprint arXiv:1812.11800.

Belbahri, M., Sari, E., **Darabi, S.**, Nia, V. P. (2019, August). Foothill: A quasiconvex regularization for edge computing of deep neural networks. In International Conference on Image Analysis and Recognition (pp. 3-14). Springer, Cham.

Kachuee, M., **Darabi, S.**, Fazeli, S., Sarrafzadeh, M. (2019). Group-Connected Multilayer Perceptron Networks. arXiv preprint arXiv:1912.09600.

CHAPTER 1

Introduction

With the ubiquity of sensors and improving hardware, large amounts of data are being collected that can be leveraged to extract valuable knowledge for many applications. To make use of these troves of data, recent deep learning methods have demonstrated tremendous success in learning from the raw input data without human expert or domain knowledge as required in traditional machine learning [1]. Applications such as image classification [2], and language translation [3] has seen enormous success through the use of deep supervised learning frameworks. An application domain of interest for extending the deep learning framework is healthcare. This can be a significant driving force for improving patient quality of care and overall quality of life.

Clinical datasets are collected as part of routine health practice across a variety of healthcare institutions. These contain both heterogeneous structured data modalities such as text and images as well as unstructured data modalities such as demographic information, diagnoses, and laboratory results. These data sources can provide vital information for making better clinical decisions on a patient's prognosis as they are undergoing care. The measurements on each patient vary based on their diagnosis and symptoms presented. Further, it is typically the case that information on each patient is incomplete; that is, not all possible measurements are made but only a subset determined by a medical practitioner. As such, patients with varying disorders may differ on multiple clinical measurements and their response to treatments, making modelling patient trajectories a challenge.

With the availability of medical datasets and deep learning frameworks, many opportunities arise for developing data-driven clinical decision support systems. However, compared to images and text, health-care datasets suffer from challenges which researchers need to address. There are many challenges to deploying an end-to-end machine learning pipeline for personalized medicine from a computational standpoint. In this dissertation, the primary focus is on limited data and the multi-modality challenges present in such datasets.

Multi-modality refers to multiple data sources available for inference as provided in Electronic Health Records (EHR). Research in this area tries to identify latent patterns from these data sources amongst cohorts of patients that can contribute to their personalized prognosis as they are undergoing care. Chapter 1 presents a method to learn unsupervised patient representation from EHR considering clinical codes and clinical texts and handling the longitudinal component of these data sources, showing it is a utility for various downstream prediction tasks. Similarly, In Chapter 2, an unsupervised method to learn patient status representation from clinical codes and signals is presented using two auto-encoding steps that cross-mix the representation from the data sources into a single representation useful for downstream prediction tasks.

Clinical datasets generally suffer from limited labelled data, where only a few examples contain human expert labels that can be employed in supervised learning. The lack of labels is due to the high cost and time required to acquire such labels from medical doctors. In such scenarios, naively employing deep learning models via supervised learning will lead to overfitting and poor generalization. Large amounts of readily available unlabeled data in the health domain can be leveraged instead to alleviate the limited data problem. Deep unsupervised methods have been previously studied in such settings to extract general-purpose representations in images or text [4, 5]. In the health domain, most datasets can be considered as tabular, where a set of rows (examples) and columns (features) that may be permutation invariant constitute the data. In Chapter 3, a semi-supervised is introduced for tabular datasets with a limited number of samples. Lastly, Chapter 4 a simple framework for dealing with a high-class imbalance in the tabular domain.

CHAPTER 2

Patient Electronic Health Record Representation

Effective representation learning of electronic health records is a challenging task and is becoming more important as the availability of such data is becoming pervasive. The data contained in these records are irregular and contain multiple modalities such as notes, and medical codes. They are preempted by medical conditions the patient may have, and are typically recorded by medical staff. Accompanying codes are notes containing valuable information about patients beyond the structured information contained in electronic health records. We use transformer networks and the recently proposed BERT language model to embed these data streams into a unified vector representation. The presented approach effectively encodes a patient's visit data into a single distributed representation, which can be used for downstream tasks. Our model demonstrates superior performance and generalization on mortality, readmission and length of stay tasks using the publicly available MIMIC-III ICU dataset. Code avaialble at https://github.com/sajaddarabi/TAPER-EHR

2.1 Introduction

Electronic health records (EHR) are commonly adopted in hospitals to improve patient care. In an intensive care unit (ICU), various data sources are collected on a daily basis as preempted by medical staff as the patient undergoes care in the unit. The collected data consists of data from different modalities: medical codes such as diagnosis which are standardized by well-organized ontology's like the International Classification of Disease (ICD)[1] and medication codes standardized using National Drug Codes (NDC)[2]. Similarly, at various stages of the patient's care physicians input text noting relevant events to the

[1]http://www.who.int/classification/icd/en
[2]http://www.fda.gov

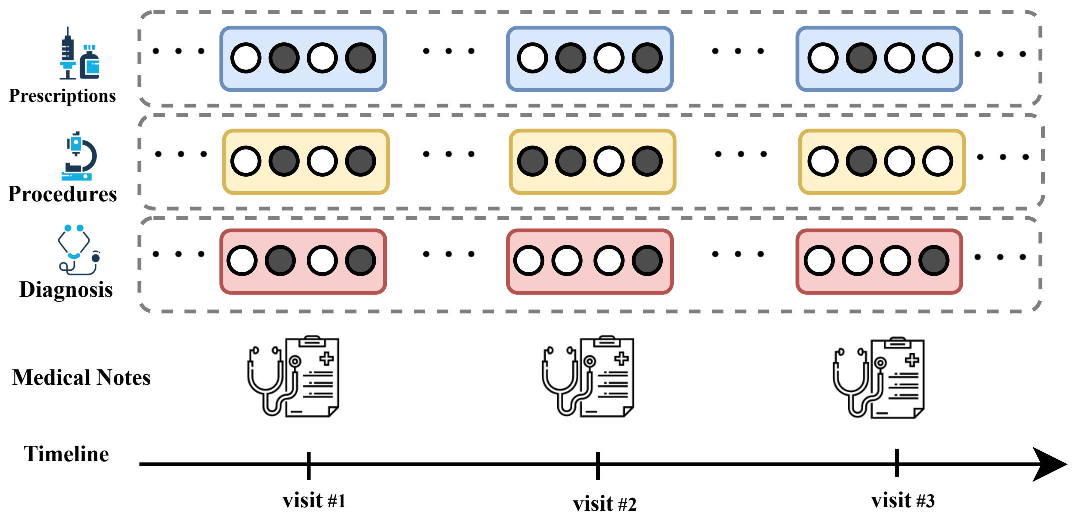


Figure 2.1: Patient timeline during an ICU visit where different data points are collected. These include prescriptions, diagnosis codes, procedure codes and medical notes.

patient's prognosis. Additionally, lab tests and bedside monitoring devices are used to collect signals each of which are collected at varying frequencies for a quantitative measure of the patient care. There is a wealth of information contained within EHRs that has a significant potential to be used to improve care. Examples of inference tasks using such data include estimating the length of stay, mortality, and readmission of patients [6, 7].

The traditional approach for healthcare analysis has mainly focused on classical methods for extracting hand-engineered features and designing rule-based systems. More recently, deep learning has demonstrated state of the art results on a varying set of tasks, in which learning intermediate representation is at the heart of all these analysis [8]. This representation can be obtained without domain-specific expertise by leveraging available EHR data. Although such methods have demonstrated great performance on image, audio datasets, leveraging deep learning techniques on healthcare data present new challenges as the data entered are sparse and contain different modalities.

As is common in natural language processing tasks, the typical method for embedding medical codes and text could be through the use of one-hot vectors; though these are naturally high-dimensional and sparse resulting in poor performance. To alleviate this, the idea of learning a distributed representations as applied to natural language processing [9] has been also applied on medical data [10]. Such methods share a common intuition that similar medical codes should share a similar context. Additionally, codes have varying temporal context, as such patients may have multiple visits with a similar set of codes. As an example, flu is short-lived whereas a diagnosis code for a more terminal disease such as cancer has a longer scope and hence will be present on all of the patient's visits. Due to the varying temporal context, it is also important to take into account the temporal scope of codes and texts assigned [11]. This demands for a model which takes the sequential dependencies of the patient's visits into account.

To capture the sequential dependencies present in medical data recurrent neural networks (RNNs), and Long Short-Term Memory (LSTM) have become the go-to model. RNN auto-encoder models are commonly augmented with attention mechanisms allowing the model to attend to specific time steps either through soft/hard attentions resulting in improvement and interpretability in the final representation obtained [12, 13]. In NLP tasks such attention mechanisms are not required to be causal in time and hence can attend to both past representation as well as future representations to generate the current representation. However, in a healthcare setting, it is desirable to have the representation be causal in time as clinical decisions are made sequentially. Recently, transformer models [14] were proposed for natural language processing tasks, and have shown impressive results. It uses self-attention and as the model creates intermediate representations of the input it attends to its representation at previous and future timesteps when considering the present representation.

Majority of patient representation work has solely focused on embedding medical codes or text as a patient representation for downstream tasks but not both. To address this, we study the use of transformer networks to embed structured medical code data as well as a language model to embed the text portion of visits. In this work, we propose to combine the medical representation from text and medical codes into a unified representation which can then be used for downstream prediction tasks. Lastly, the presented study takes into account the temporal context of a visit and embeds subsequent visits given the patient's history. In the following sections, we briefly go over related works and present our method followed by experimental setup and results.

We have made our code and preprocessing steps available[3].

2.2 Related Works

The idea of learning embeddings for sparse one-hot vector data types using back-propagation was presented in [15]. Follow up work learned these embedding using neural networks *Bengio et al.*[16]. Since its success in NLP *Mikolov et al.*[9] for language modeling, similar approaches have been used in the health domain. The two intuitive methods that are commonly used for word embedding are (1) skip-gram where a current word is said to predict surrounding words, (2) continuous bag of words (CBOW) where a set of words are made to predict a center word. In *Choi et al.* [17] medical codes (diagnosis, procedure, and medications) are concatenated as one-hot vectors and embedded using the skip-gram model. The intuition behind skip-gram model is: codes in a visit should be predictive of its surrounding immediate patient visit codes as well. They also present an additional code loss term as regularization to the objective. It follows the intuition that codes in a visit should also predict one another. Similar to [17], *Nguyen et al.* [18] use the concatenation of code representation and apply 1D convolutional network to obtain a visit representation. Follow up work augment these methods with external ontology's and attention on such external data sources when learning the representation [19, 20].

More recent work takes advantage of the hierarchical structure present in medical codes as they are assigned to a patient. For example in [21], the final visit representation is created hierarchically: first codes are embedded at a treatment level where a set of medication/procedures codes predict diagnosis codes, followed by diagnosis level where the representations at this level are made to predict next visits codes. Empirically the method can learn from a small set of samples and outperform earlier methods as presented on their proprietary datasets. Although these methods achieve reasonable results they do not explicitly model temporal context. This is important as certain clinical codes are short-lived

[3]https://github.com/sajaddarabi/TAPER-EHR

whereas others could be long-lived or be permanent. As a result, certain codes should not be regarded as the context of one another although they may occur in the same visit or subsequent visit. This has been studied in *Cai et al.*[11] where they train a CBOW model with temporal attention on the code representations. Most of these methods are concerned with medical code embedding and disregard physician notes which can potentially attribute to an improved representation.

Medical notes contain a vast amount of information but have not been studied adequately, especially for downstream tasks in the medical setting. Most works have focused on clinical concept embedding instead. In a clinical setting, nurses and doctors document patients progress. As notes are typically recorded using medical jargon, they do not necessarily follow the common grammatical structure found in English text. As such building, a representation from medical text using hand-engineered features is a challenge [22]. For example in [23] the authors use unstructured EHR data and learn semi-supervised patient representation which are then evaluated on downstream tasks such readmission, mortality and length of stay. Along this line of work, in [24] text from EHR are used to embed patient text by predicting billing codes and averaged for downstream tasks using neural networks. Similarly, in [25], the authors evaluate different models for embedding clinical notes such as CNNs, LSTMs and evaluate them on chronic disease prediction. Although they showed good results their models are not expressive enough to capture all of the salience present in clinical text. To this end, a recent model namely Bidirectional Encoder Representation Transformers (BERT) language model presented by *Devlin et al.*[5] have recently outperformed previous methods on many benchmarks. This model was used in [26] to embed medical notes of patients. They evaluate their representations predictive performance on downstream tasks showing state of the art results compared to other methods.

Few works have studied the combination of both text and clinical codes. Previous work [27], trains skip-gram and word2vec models to jointly embed clinical concepts and clinical text into a unified vector. Similarly, in [28] they use clinical text to predict clinical concepts. Although both text and code are taken into account, they are different from transferring the joint representation to downstream tasks that is the focus of our work.

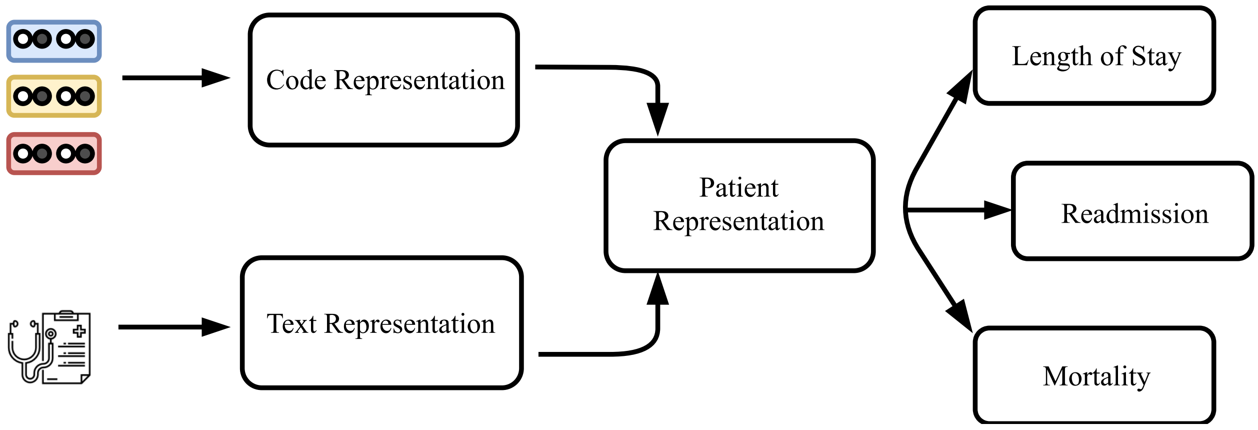


Figure 2.2: Overview of method used to obtain patient visit representation.

2.3 Method

The objective of this work is to create a distributed representation for a patient based on text and medical codes. This representation is then fed to classifier for predictive analytics tasks such as mortality, length of stay, and readmission (Fig. 5.1). We split the training into two steps, (1) Skip-gram model using transformer networks to learn medical code representation, (2) a BERT model is trained on medical notes and the resulting representations at a time step are summarized using auto-encoder architectures [29, 30, 31]. The final representation for a patient is a concatenation of these two. We discuss the approach in more detail in the following subsections.

To present the problem setting, the sequence of EHR data under consideration consists of a finite set of medical concepts $\mathcal{C} = \mathcal{M} \cup \mathcal{D} \cup \mathcal{P}$, where $\mathcal{M}$ is the set of medication codes, $\mathcal{D}$ is the set of diagnosis codes, and $\mathcal{P}$ is the set of procedure codes. Accompanying the codes are medical notes $\mathcal{T}$. We denote a patients longitudinal data as $\mathcal{D}^T = \{(c_0, t_0), \cdots (c_T, t_T)\}$ with T visits where c_i and t_i correspond to the codes and texts assigned respectively within the same visit window.

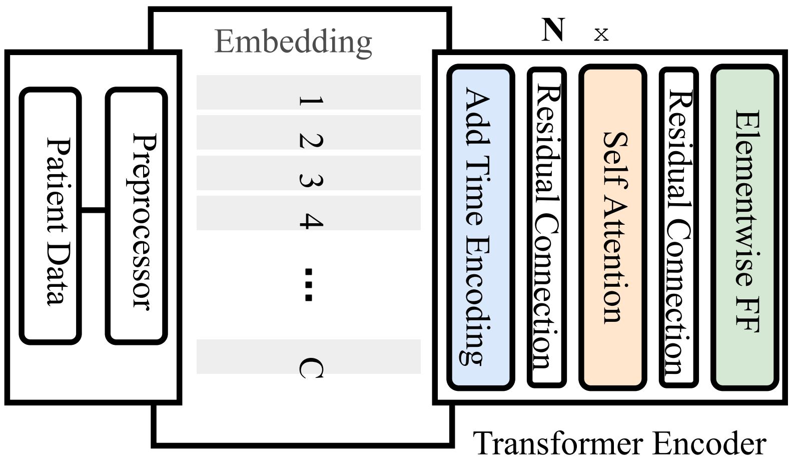


Figure 2.3: The code representation module is a transformer encoder, which takes as input patient clinical codes. The embedding matrix is a $\Re^{d\times C}$ matrix. Clinical codes are embeeded using the embedding matrix, which is then passed to the transformer encoder block.

2.3.1 Medical Code Embedding

We use skip-gram model to learn code embeddings as it is able to capture relationships and co-occurence between codes. We briefly review skip-gram model presented by *Mikolov et al.*[9]. Given a sequence of codes $\{c_1, c_2, \ldots, c_T\}$, where each code vector is a binary vector $c_t \in \{0,1\}^{|\mathcal{C}|}$, the model is tasked to predict the neighbouring codes given a code c_t. The objective can be written as

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-w\leq j\leq w, j\neq 0} c_{t+j}^{\top}\log(\hat{c}_t) + (1-c_{t+j}^{\top})\log(1-\hat{c}_t) \tag{2.1}$$

Here w is the context window, and the softmax function is used to model the distribution $p(c_{t+j}|c_t)$. We use multiple transformer encoder layers with self-attention mechanism as the model for skip-gram. This model is then trained on medical code sequence $S = \{c_0, \cdots c_T\}$ by stacking code vectors into a matrix $K \in \{0,1\}^{T\times|\mathcal{C}|}$. The resulting set of codes are then converted into a set of embedding codes $e_{c_t} \in \Re^d$ using an embedding matrix $W \in \Re^{|\mathcal{C}|\times d}$. The embedding for the set of codes c_t at visit t is obtained as

$$e_{c_t} = W^T c_t \tag{2.2}$$

As the model does not contain any recurrence or convolution, to enable the model to make use of the ordering we need to inject information about the relative positioning of each embedding. This is done by adding to each embedding position a sinusoidal with frequency as a function of its timestamp t as suggested by the original transformer network. This signal acts as positional-dependent information which the model could use to incorporate time. The model is summarized in Fig. 2.3. We stack multiple transformer layers following on top of the embedding matrix. By transformer layer, we mean a block containing the multi-head self-attention sub-layer followed by feed-forward and residual connections. For more details on this refer to [14] and the *tensor2tensor* library[4]. As multi-head attention can attend to future time steps, to ensure that the model's predictions are only conditioned on past visits, that is embedding at time step t can only attend to previous time steps $t-1, t-2\ldots$, we mask the attention layers with a causal triangular mask. This is the same "masked attention" in the decoder component of the original transformer network. This mask is applied to the set of embedding

$$E = \{e_{c_1}, e_{c_2}, \ldots e_{c_T}\} \tag{2.3}$$

$$\mathit{Attention}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d}})\boldsymbol{V} \tag{2.4}$$

in the encoder block to ensure causality. Where the query Q, key K, and value V are set to the sequence of embeddings E, and d is the embedding dimension. To obtain the final code representation for timestep t, the t^{th} output of the self-attention output is used. We call this code representation E_{c_t}.

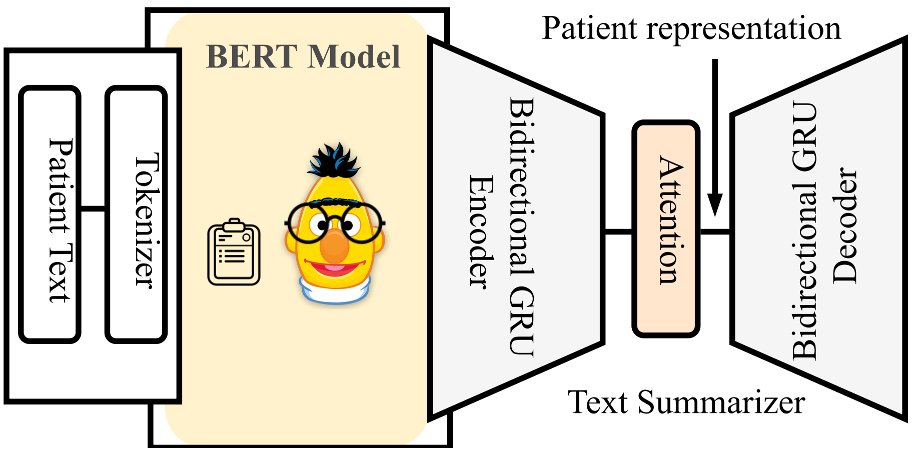


Figure 2.4: The text representation module takes as input patient text and pre-processes them into tokens, which are embedded using BERT. Subsequently, it is fed to a text summarizer autoencoder network. In this work, a LSTM-AE is used to learn the intermediate patient representation.

2.3.2 Medical Text Embedding

An overview of the text module is shown in Fig. 2.4. The embedding for the medical text sequence of a patient for time sequence $\{t_1, t_2, \ldots, t_T\}$ is obtained by using a pre-trained BERT model initialized from BioBERT [32] followed by a bidirectional GRU as a text summarizer. This is done, as the pre-trained BERT model has a fixed maximum sequence length of n limiting the sequential scope of the text. Further, medical notes could get very lengthy during a visit and they contain different types of notes such as nurse notes, pharmacy notes, discharge notes, etc. the aggregate length of these at a time step t could surpass the fixed length size of n. As the aim is to obtain a single visit representation, the aggregate notes at until time step T are batched into a set of sentences $(u_1, u_2, \cdots, u_m)$, where $u_i \in \mathcal{Z}^n$ and m is the maximum occurring length in the corpus after batching each visits text into sentences of n words. The resulting set of sentences for a visit is embedded

[4]https://github.com/tensorflow/tensor2tensor

using the BERT model which results in a matrix $U \in \Re^{m \times d_{BERT}}$.

$$h_{1:m} = GRU_{enc}(U_m, h_0) \tag{2.5}$$

$$E_{U_t} = \text{softmax}(\frac{\boldsymbol{h_{1:m}h_{1:m}^T}}{\sqrt{d^{text}}})\boldsymbol{h_{1:m}} \tag{2.6}$$

$$\hat{U} = GRU_{dec}(U_m, h_m) \tag{2.7}$$

$$L(U_{1:m,t}, \hat{U}_{1:m,t}) = \sum_{i}^{m}(u_i - \hat{u}_i)^T(u_i - \hat{u}_i) \tag{2.8}$$

Subsequently, the set of sentence representations are summarized into a single patient text representation using a text-summarizer module. This module follows an auto-encoder architecture with GRU's as the building block. The input is a set of sentence representations $\in \Re^{d_{text}}$ obtained by the BERT module applied on the aggregate text until time t followed by a self-attention head on the hidden representations, where the decoder is tasked to output a sentence representations $\in \Re^{d_{text}}$ following the bottleneck. The objective of the summarizer is to reduce the MSE loss objective between the input sequence of the text embeddings and the models predicted representation at the correspondingly same time-step. Finally, the patients visit text representation is obtained by summarizing the set of sentence representations $\{U_1, U_2, ..., U_m\}$, where the output of the attention head applied on the encoders hidden representations is used as the representation.

2.3.3 Patient Representation

The final patient representation Z^t at time t is obtained by concatenating the code, and text representations. Additionally, the demographics d_t of the patient recorded in visit at time t is concatenated to the resulting vector. The demographics of patient contain information such as age, gender, race, etc, where categorical values are coded as one hot vectors. The final representation is denoted as $Z^t = [E_{c_t}; E_{U_t}; d_t]$, where the size of this vector is the sum of the components $d_{embedding} + d_{enc} + d_{demographics}$. This representation is used for downstream tasks.

We provide specific values for the dimensions of each component in our implementation details in the experiments section.

2.4 Experiments

2.4.1 Dataset

We evaluate our model on the publicly available MIMIC-III clinical Database [33]. It consists of EHR records of 58,976 hospital admission consisting of 38,597 ICU patients from 2001 to 2012. On average, each patient has 1.26 visits. The database contains tables associated with different data, where we extracted demographics, medical codes, and medical notes.

2.4.1.1 Readmission task

The first task is to predict 30-day unplanned readmission to the ICU after being discharged. In this task, we formulate it as a binary problem, that is to predict whether a patient will be readmitted within 30-days after being discharged. Text entered into the MIMIC database, contains different reports, such as nurse notes, lab results, discharge summaries etc. We limit the text for each visit to contain the discharge summaries or text entered within the last 48h before the patient is discharged in the absence of discharge summaries.

2.4.1.2 Mortality task

The second task is to predict mortality of patient, whether they passed away after being discharged or within the ICU. Similar to readmission it is formulated as a binary task. In this task, mortality related codes are discarded from the dataset and patients who were admitted for organ donations are removed. Additionally, the input text for each visit is limited to the first 24h of the admission.

2.4.1.3 Length of stay task

: The third task is to forecast the length of stay (LOS) for patients. In this task, longer LOS is an indication of more severe illness and complex conditions. We formulate this problem as a multi-class classification problem by bucketing the length of stay into 9 classes: 1-7 correspond to one to seven days respectively, 8 corresponds to more than 1 week but less than 2, 9 corresponds to more than 2 weeks. The model is tasked to predict $P(Y = L|Z_t)$ where $L \in 1, 2, 3, 4, 5, 6, 7, 8, 9$ denoting the different time intervals defined previously.

2.4.1.4 Code prediction task

: In this task, clinical codes are predicted for new admissions of patients given past clinical codes and historical data of the patient. The predicted vector is high-dimensional equal to the size of unique codes.

2.4.2 Data preprocessing

We extracted procedure, and diagnosis codes for each patient visit. These codes are defined by the International Classification of Disease (ICD9) and medications using the National Drug Code (NDC) standard. The total number of ICD9 codes in MIMIC-III is 6984, the number of drug codes is 3389, and the number of procedure codes is 1783. Codes whose frequency are less than 5 are removed. We used the Clinical Classification Software for ICD9-CM[5] to group the ICD9 diagnosis codes into 231 categories. The Clinical Classification Software for Services and Procedures[6] was used to group the procedure codes into 704 categories. Additionally, patients of age under 18 were removed from the cohort. As medical notes contain many errors, we correct grammatical errors and remove non-alphanumerical characters. The text preprocessing closely follows [26]. After preprocessing, the average recorded number of codes per visit is 20.52, the average number of words in medical notes is 7898 and the average number of visits per patient is 1.29. The statistics of the compiled cohort are depicted in Fig.2.5

[5]https://www.hcup-us.ahrq.gov/toolssoftware/ccs/ccs.jsp
[6]https://www.hcup-us.ahrq.gov/toolssoftware /ccs_svcsproc/ccssvcproc.jsp

2.4.3 Experimental Setup

2.4.3.1 Model Configuration & training details

The training follows the method discussed in Section 5.3. A medical concept model is trained independently on clinical codes in an unsupervised manner and similarly, the text summarizer is trained on the text portion.

To train the transformer encoder we explored different values for hyperparameters, namely number of layers, number of multi-head attention heads n_{head}, dimension for each head d_{head} and the final model representations d_{code}. We found 2 layers perform well. We set the models representation dimension (d_{code}) to 128. Further, the self-attention module contains 8 head (n_{head}), each with dimension 64 (d_{head}), which are a common configuration used in transformer networks. The network is trained using Adam [34] with a cosine annealing schedule and with a period of 50 epochs. The initial learning rate is set to 0.00025. The window size for the skipgram objective is set to 2.

We initialize the BERT model with the pre-trained weights on medical nodes as presented in [35]. In this work, the BERT language model is initialized with BioBERT which is a model that has been trained on a large corpus of public medical data such as PubMed, medical abstracts, etc. Then the model was fine-tuned on the MIMIC-III clinical notes.

To train the text summarizer we use a 2-layer bidirectional GRU autoencoder with the intermediate representation set to $d_{enc} = 128$. A teacher-forcing ratio of 0.5 is used with a step learning rate schedule decay of 0.1 every 50 epochs with initial lr set to 10^{-3} [36]. We have also tried using cosine annealing schedule, though this did not result in improvements.

Lastly, the classifier for downstream tasks is a simple 2-layer fully connected network. The first layer contains $\frac{d_{code}+d_{text}+d_{demographics}}{2}$ neurons with ReLU activation followed by a layer which maps to number of classes in the downstream task. When training on downstream tasks only the classifier weights are trained for 30 epochs with a step learning rate schedule decay of 0.1 every 10 epochs. This setting is used for all downstream tasks.

2.4.3.2 Implementation details

We implemented all the models with Pytorch 1.0 [37]. For training the models we use the Adam optimizer [34]. In all experiments, the batch size is set to 32 on a machine equipped with 1 NIVIDIA 1080TI CUDA 9.0, 32GB Memory & 8 CPU cores.

2.4.4 Evaluation Metrics

The compiled cohort consists of patientuids as keys that are unique to each patient which are used to create the test/train split. This is done using k-fold with $k = 7$ on the patient keys, which the unsupervised models are trained on the train portion and validated on test ($\approx 15\%$ of total data). The output for a particular time step is evaluated using the patient representation Z_t at time t.

2.4.4.1 Area under the precision-recall (AU-PR)

this metric is the cumulative area under the curve by plotting precision and recall while varying the outputs $P(y_t = 1|Z_t)$ true/false threshold from 0 to 1.

2.4.4.2 Receiver operating characteristic curve (AU-ROC)

this metric is the area under the plot of the true positive rate against false positive rate while varying outputs $P(y_t = 1|Z_t)$ true/false threshold from 0 to 1.

2.4.5 Baselines

We compare our model with the following baselines

- **Med2Vec** [17]:
  A multi-layer perceptron is trained on medical codes using the skip-gram objective function on a visit basis. An additional loss term is used for the co-occurrence of codes within the same visit as a regularization. The resulting output is a set of code

representations in $\Re^d$.

- **ClinicalBERT** [26]:

  A BERT model is pre-trained on public medical data, which is then fine-tuned on clinical text. Following this pre-training a BERT classifier is initialized with pre-trained weights and further fine-tuned on downstream tasks. The input to this network is text.

- **Time-aware Embedding** [11]:

  A multi-layer perceptron is trained on medical codes with an additional attention layer to take into account the temporal context of medical codes. The resulting model is trained using either skip-gram/CBOW.

- **Patient2Vec** [20]:

  In this work a sequence of medical codes are embedded using word2vec model. The sequence of visits with irregular time intervals is then binned into a set of subsequences with standard intervals. Subsequently, the embedded vectors are stacked into a matrix where convolution stacked with GRU and attention models are applied to obtain the final patient representation.

- **Joint-Skipgram**[27]:

  The embeddings are trained using both text and code as the vocabulary. In addition to the traditional skipgram loss, i.e. codes in the same visit predicting surrounding codes or text predicting surrounding text; the skipgram objective is modified such that text in a visit predict codes in the same visit and vice versa.

- **Deepr**[18]:

  A set of clinical codes are embedded using skip-gram model. As visits contain multiple codes, the vectors corresponding to each code is stacked into a matrix, then the set of matrices for each visit is fed to a convolutional neural network and max-pooling layers to extract the final patient representation.

- $Sg_{code} + Sg_{text}$:

  Embeddings for both code and text are learnt using the skipgram objective independently. Subsequently for downstream tasks a patient representation is obtained by concatenating the code and text embeddings.

- **Supervised**:
  The BERT model and summarizer takes as input raw text and transformer model raw codes, which is trained jointly on downstream tasks without pretraining.

We do not compare with more traditional text embeddings such bag of words (BOW) as other work have shown the benefits of using BERT as text representation in NLP tasks.

We study the effect of different components presented by adding/removing text/code/demographics representation to our final patient visit representation.

Table 2.1: Code Recall

Method	Diagnosis@k				Procedure@k			
	10	20	30	40	10	20	30	40
Ours	47.92%$_{\pm(0.6)}$	65.11%$_{\pm(0.4)}$	75.25%$_{\pm(0.3)}$	82.00%$_{\pm(0.4)}$	50.99%$_{\pm(1.2)}$	62.25%$_{\pm(0.9)}$	67.97%$_{\pm(1.1)}$	71.52%$_{\pm(1.0)}$
Joint-Skipgram [27]	41.16%$_{\pm(0.6)}$	59.49%$_{\pm(0.7)}$	71.26%$_{\pm(0.7)}$	79.73%$_{\pm(0.5)}$	50.88%$_{\pm(0.9)}$	61.33%$_{\pm(0.7)}$	67.32%$_{\pm(0.9)}$	70.95%$_{\pm(0.7)}$
Med2Vec [17]	42.24%$_{\pm(0.2)}$	60.18%$_{\pm(0.3)}$	70.89%$_{\pm(0.4)}$	78.42%$_{\pm(0.4)}$	47.44%$_{\pm(0.7)}$	58.10%$_{\pm(0.6)}$	64.19%$_{\pm(0.6)}$	68.36%$_{\pm(0.7)}$
MCE [11]	45.70%$_{\pm(0.4)}$	63.43%$_{\pm(0.4)}$	74.44%$_{\pm(0.4)}$	81.14%$_{\pm(0.4)}$	51.87%$_{\pm(0.8)}$	61.62%$_{\pm(1.0)}$	68.42%$_{\pm(0.8)}$	71.97%$_{\pm(0.8)}$
Deepr/P2V [18, 20]	35.30%$_{\pm(0.3)}$	52.40%$_{\pm(0.4)}$	65.04%$_{\pm(0.4)}$	74.14%$_{\pm(0.3)}$	42.47%$_{\pm(0.7)}$	55.59%$_{\pm(1.1)}$	62.42%$_{\pm(0.8)}$	66.49%$_{\pm(0.9)}$
Skipgram	35.23%$_{\pm(0.4)}$	52.30%$_{\pm(0.4)}$	65.08%$_{\pm(0.4)}$	74.13%$_{\pm(0.2)}$	42.56%$_{\pm(0.7)}$	55.81%$_{\pm(1.0)}$	62.61%$_{\pm(0.9)}$	66.70%$_{\pm(0.9)}$

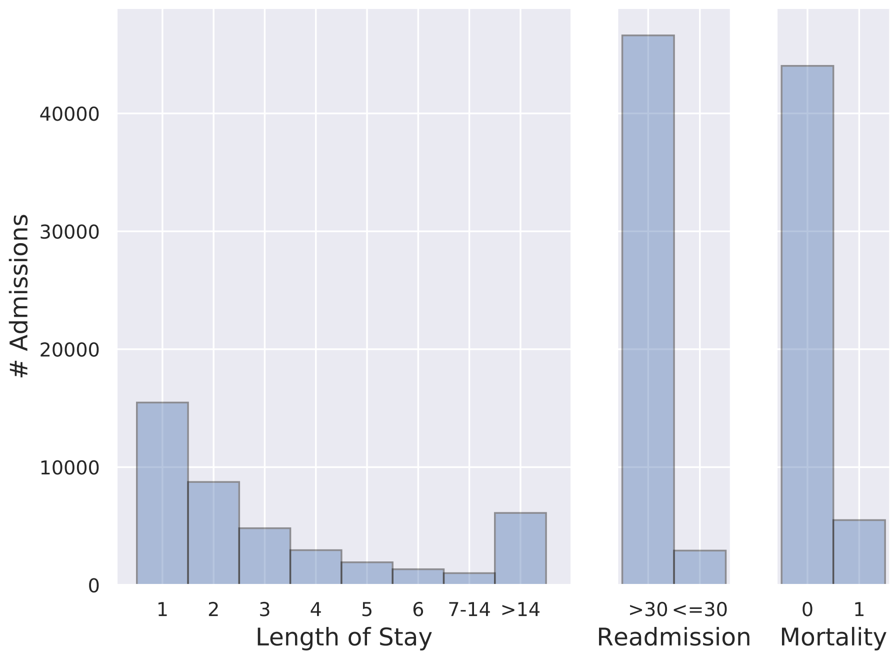


Figure 2.5: Statistics of compiled cohort for both length of stay and readmission downstream tasks.

2.5 Results

To show the expressiveness of our representation, we evaluate its performance compared to baseline methods on downstream tasks and unsupervised learning tasks presented in the experiment section. We present the results obtained by embedding both text and code as patient representation on three downstream tasks: (1) 30-day readmission, (2) mortality, and (3) length of stay (LOS). Note that in MIMIC-III database clinical codes are entered into the database upon discharge of a patient, as consequently the presented results may not be immediately clinically actionable in this case. Although, this may not be the case in other datasets where codes are updated throughout a visit. To this end the codes on the same visit are not fed to the model, but rather the previous set of codes are used.

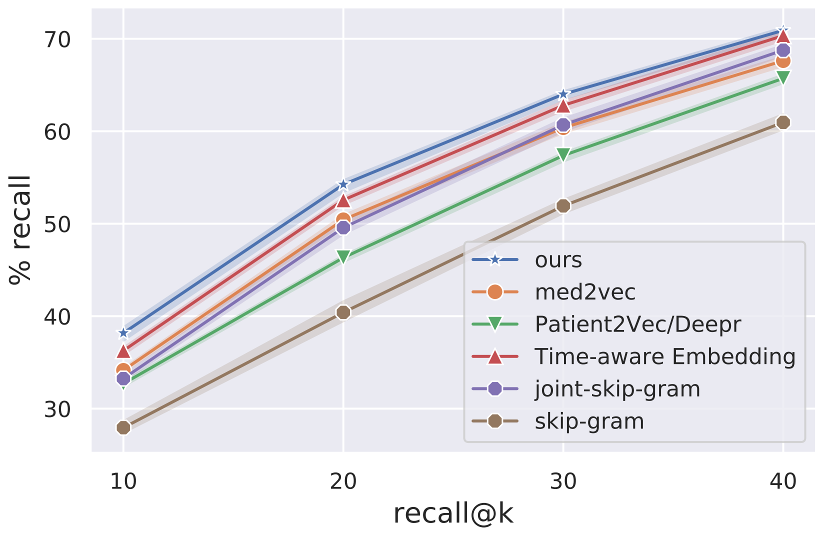


Figure 2.6: Performance of different embedding schemes on next visit code prediction using recall @k.

2.5.1 Code pre-training

The encoder network is trained on clinical concepts and as most patients have 3 hospital visits after filtering to atleast 2 visits the window size for the loss term is set to 2. The performance of our network is compared with baseline methods using recall@k. This metric is evaluated by computing

$$\text{recall}@k = \frac{\#\text{ of relevant codes in top k}}{\#\text{ number of relevant code}}$$

This metric mimics a practitioners method of arriving at a diagnosis or prescribing medications where they generally have several sets of candidates as a presumed cause for the underlying condition of the patient. As shown in Fig. 2.6 our code embedding consistently outperforms the baselines. All baselines are fine-tuned on the same corpora by exploring different architectural hyper-parameters, except the embedding size which is fixed to $d_{code} = 128$ for all models. A comparison on recall for different codes (diagnosis, procedure) is also reported in Table. 2.1. From the results the time-aware code representations outperform other baselines.

2.5.2 Ablation Study

To evaluate the different components of the proposed method we conduct an ablation study on the inclusion/exclusion of the components in the final representation for downstream tasks. The complete representation using demographics, text representation and clinical code representation is the concatenation of these on a visit $Z^t = [E_{c_t}; E_{U_t}; d_t]$.

2.5.2.1 Readmission

To better evaluate the effect of the different embedding components, we run an ablation study. The results are reported in table 2.2.

Table 2.2: Downstream Tasks: Readmission

Method	AUC-ROC	PR-AUC
Text+Code+Demo	67.42%	68.03%
Text+Code	65.74%	65.43%
Text+Demo	61.44%	62.81%
Code+Demo	64.53%	67.68%
Text	56.44%	56.55%
Code	60.74%	57.89%
Demo	54.76%	59.01%

From the results, it can be seen both text and code are informative for classifying readmission. The complete combination of text, code, demographics outperforms others in this case.

2.5.2.2 Mortality

Mortality task is concerned with predicting whether a patient will pass away within a pre-defined window. We predict mortality on a visit basis i.e. does the patient pass away in the current visit to the ICU. An ablation is done in table 2.3. Similar, to previous ablations, the combination of text, code, and demographics outperforms other combinations.

Table 2.3: Downstream Tasks: Mortality

Model	AUC-ROC	PR-AUC
Text+Code+Demo	63.42%	65.65%
Text+Code	59.75%	60.33%
Text+Demo	61.54%	64.07%
Code+Demo	60.41%	64.41%
Text	57.14%	57.72%
Code	56.91%	53.71%
Demo	60.10%	62.02%

2.5.2.3 Length of Stay

In general length of stay is a much more challenging task compared to readmission binary task. In this classification task, we limit the medical text to the first 24-hours of the current patient visit in which length of stay is being predicted. Limiting the note context window is done as medical text could include information on date patient has been discharged. In this task the network is trained on imbalanced class data split and tested on balanced data, this is done as the majority of classes are discharged within 24h-48h.

As shown in table 2.4, the combination of text, code and demo outperforms others. We

Table 2.4: Downstream Tasks: Length of Stay Top-1 avg over 5-fold cross validation

Method	Top-1
Text+Code+Demo	25.57%
Text+Code	23.38%
Text+Demo	21.10%
Code+Demo	22.22%
Text	18.82%
Code	20.54%
Demo	20.13%

conclude in this set of experiments, solely using text and diagnosis codes is not predictive enough to forcast the length of stay of patients.

2.5.2.4 Comparison with Other Work

Table 2.5: Comparison with other work on downstream tasks.

Method	Readmisison		Mortality		LOS
	ROC	PR-AUC	ROC	PR-AUC	ACC
Ours	67.42%±(2.1)	68.03%±(1.5)	63.42%±(0.6)	65.65%±(0.9)	25.57%±(2.7)
ClinicalBert [26]	64.41%±(2.3)	67.71%±(1.3)	61.23%±(1.9)	64.83%±(1.1)	22.65%±(1.6)
Joint-Skipgram* [27]	64.59%±(2.9)	65.27%±(2.7)	61.73%±(1.6)	65.03%±(1.1)	23.27%±(1.6)
SG_{code} + SG_{text}*	65.79%±(1.9)	66.72%±(0.6)	62.23%±(0.6)	64.77%±(1.1)	24.54%±(0.4)
MCE* [11]	64.61%±(2.1)	63.96%±(2.1)	62.72%±(2.1)	64.78%±(2.2)	22.47%±(2.3)
Deepr* [18]	62.63%±(1.9)	62.37%±(1.5)	60.49%±(1.2)	61.45%±(1.5)	22.81%±(0.5)
Patient2Vec * [20]	56.13%±(3.3)	65.12%±(1.1)	61.19%±(1.4)	62.51%±(1.6)	21.46%±(1.2)
Med2Vec [17]	63.44%±(2.8)	62.77%±(2.2)	61.67%±(1.6)	63.22%±(0.8)	24.15%±(0.4)
Ours - Supervised	65.38%±(3.4)	66.07%±(2.4)	62.26%±(1.7)	64.32%±(2.3)	25.12%±(3.2)

1. Methods marked * is our best-effort re-implementation due to unavailability of source.

We ran all baseline models on three downstream tasks using the same pre-processing steps. The final results are reported in table 2.5. To compare our method, we use the combination of text, code, and demographics. As demographics has proven to have predictive power for the downstream tasks, other methods are augmented to use this as input/concatenated to the output representation prior to prediction. From Table. 2.5 the presented method outperforms others by a 1-2% margin, on all tasks demonstrating the usefulness of unsupervised pre-training.

2.6 Conclusion

Effective representation learning for EHR data is an essential step to improving care. We study embedding both medical codes and notes into a unified vector representation for downstream task prediction. The presented method effectively takes the temporal context of these two data streams and provides a patient visit representation. The proposed method was evaluated on three tasks namely, readmission, mortality, and length of stay outperforming other methods. An ablation study was also done showing the usefulness of both text and code when modeling patient visits. Future work could focus on adding additional data streams to the pipeline by taking into account real-time vitals and measurements taken from a patient as they undergo care.

CHAPTER 3

EICU EHR: Unsupervised Patient Signal & Clinical Codes as Patient Status Vector

Effective modeling of electronic health records presents many challenges as they contain large amounts of irregularity most of which are due to the varying procedures and diagnosis a patient may have. Despite the recent progress in machine learning, unsupervised learning remains largely at open, especially in the healthcare domain. In this work, we present a two-step unsupervised representation learning scheme to summarize the multi-modal clinical time series consisting of signals and medical codes into a patient status vector. First, an auto-encoder step is used to reduce sparse medical codes and clinical time series into a distributed representation. Subsequently, the concatenation of the distributed representations is further fine-tuned using a forecasting task. We evaluate the usefulness of the representation on two downstream tasks: mortality and readmission. Our proposed method shows improved generalization performance for both short duration ICU visits and long duration ICU visits.

3.1 Introduction

Learning patient representation is a popular topic in health analytics. With the availability of electronic health records (EHR) systems and increasing amounts of data, it has opened avenues in learning these representations using deep learning methods. The general approach for that matter is learning representations through the use of supervision signals. Whereas in regimes, where labels are hard to acquire or not readily available unsupervised learning is used to leverage the data regardless of the presence of labels. As healthcare naturally suffers from limited labels and high costs with obtaining labels, learning reusable feature representations from large unlabeled samples has been an area of active research. Unsupervised learning can be used to produce representations of general utility, which can be further used for downstream tasks such as mortality, readmission, and length of stay.

Routine medical practice generates a wealth of patient time-series, most of which are preempted by conditions a patient may have as they undergo care. Annotating the different data inputs to the system often requires medical experts incurring high costs to label the data. In addition to this, there are challenges in processing EHR data, and applying machine learning methods on patient data is not immediately obvious due to complicating factors in the collection process. To enumerate a few challenges: 1) the data is multi-modal (Fig. 3.1). 2) in the recorded values there are many missing and incomplete values requiring pre-processing or imputation techniques. 3) there is a natural structure in the collection process that may not be captured in a single patient, but more so when looking at a body of patients, 4) Temporal nature of clinical events. Representation learning can help overcome some of these challenges.

The choice of representation plays a significant role in machine learning algorithms and their performance [8]. Many efforts are put into developing methods that would enable learning representations that generally lends itself to improved predictive performance. There are different approaches to representation learning, such as clustering, principal component analysis, independent component analysis, and, more recently, deep-learning, which has outperformed other methods [8]. Deep representation learning uses a set of non-linear transformations of the input data, which results in more abstract features as the network depth increases.

The use of unsupervised methods for learning representations is widely recognized for solving problems with limited data and label information. Once a high-level representation is learned downstream supervised tasks could become much easier for the model mapping the representations to the desired output. Further, unsupervised learning typically imposes intermediate optimizations which allow the network to learn the input distribution and learning regularities present in the input. This might be helpful in avoiding scenarios where supervised learning would poorly generalize. Perhaps the most common unsupervised method is autoencoder architectures that attempt to map the input to the same input with minimal distortion [38],[8]. Similar ideas are commonly employed in natural language processing tasks that learn word embeddings [16],mikolov2013. Recently, these methods have been employed to learn medical concepts and have shown promising results [17], [19],[11],[39].

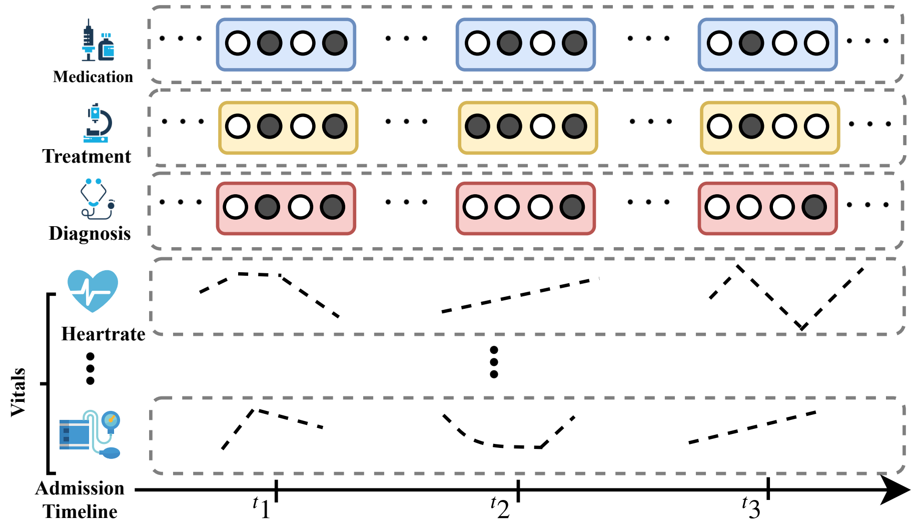


Figure 3.1: EHR patient timeline contains multi-modal sparse data each of which are entered at varying frequencies.

Previous work at large use unsupervised learning solely on clinical concepts, and disregard the patients progress throughout their ICU visit. It is not obvious how to combine other data modes to obtain a patient representation. Also, as embeddings are made to predict the surrounding context, the data used to train the embeddings are limited to patients with multiple visits, which could drastically reduce the dataset size. For medical and healthcare settings, it is essential to develop techniques that not only yield good performance on supervised tasks but are also efficient and reliable [40]. In this paper, we propose to use unsupervised learning on both clinical concepts and vital signals using networks that take into account the sequential context of a patient leveraging good portions of the data. The obtained representations can be later used as a patient's status vector for downstream task prediction.

The contributions of the paper are the following:

- We propose a two-step unsupervised fine-tuning task for embedding patient data: (1)

a single visit autoencoding step followed by a (2) forecasting autoencoding step

- We use multi-modal data, namely, clinical concepts + vital signals and show improved generalization performance for both long duration and short duration ICU stays on the eICU dataset.

The paper is organized as follows: we describe related works in section 3.1, we then present our proposed method for unsupervised pretraining in section 3.2, followed by section 3.3.2 which describes the experimental setup and the results of our proposed method in section 3.4.4.

3.2 Related Works

The unsupervised learning used in this work stems from the widely used auto-encoding principle [15],baldi2012autoencoders. Deep auto-encoder architectures are pre-trained and followed by a supervised learning phase to train a top classifier layer that is fine-tuned on specific tasks [41],[42]. These have been naturally extended to other architectures and learning methods such as variational, denoising, convolutional autoencoders[30],[43],[44]. In [9] they extended this idea to natural language processing and introduced two intuitive methods for word embeddings: (1) skip-gram where a current word is said to predict surrounding words, (2) continuous bag of words (CBOW) where a set of words are made to predict a center word [12]. As the text is naturally ordered, this idea was extended to sequential models, which are more fitting for sentences and learning sequential dependencies. For example, the popular Seq2Seq architecture is commonly used to learn language models for language generation [45]. As these methods have become widespread in Natural Language Processing, they have also been extended to medical settings.

Deep unsupervised representation learning has been successfully applied to various health settings such as EHRs (clinical concepts), medical text, and imaging data with the propose of phenotyping or classification. The general approach is to extract features using an unsupervised model and to stack it with a classifier for downstream tasks. For instance, in

[46], a stack of denoising autoencoder multi-layer perceptron was used to predict future disease codes. They run their method on a large database consisting of approximately 700,000 patients and show improved predictive performance. Similarly, in [17, 47], a multi-layer perceptron using the skip-gram model was trained with an added regularization for the co-occurrence of codes within a visit to embed the clinical concepts on MIMIC III. As EHR data might have limited occurrences of specific codes and hence hinder the learning process of deep models, later work suggested to add an attention mechanism on external ontology's [19]. More recent work on clinical concept embedding focused on leveraging temporal context using attention [11]. From the results, they suggested learning time-aware representations is critical to improving the performance for a clinical decision task. Generally, the aforementioned methods consider different flat representations from different sources and use their concatenations. Other methods leverage a hierarchical prior to the learning process [21], where medication and procedure codes are made to predict diagnosis codes enabling the model to benefit from the structure present in EHRs. They further improve on this idea by attempting to learn to have the underlying model learn the EHR structure [48]. Most of these methods are mostly limited to clinical concepts and do not use other portions of EHR data, such as vital signals.

Several works have studied unsupervised methods for clinical time series signals[49]. In [47], they use LSTM autoencoders to learn a representation for clinical signals and evaluate the overall performance of the AE architecture. Similarly, [50] preprocess signals using frequency-domain transforms and further embed these using an autoencoder bottleneck resulting in a final representation for detecting false arrhythmia alarms. More relevant work [51], use a Seq2Seq model for learning representations of clinical time series with an autoencoder loss. Following this, they assess the representation by feeding them to an LSTM classifier and evaluate on mortality/readmission tasks. They show the benefits of unsupervised learning for limited data settings compared to supervised learning based on prediction performance. In this paper, we focus on an unsupervised framework for learning a patient status vector using these data streams as inputs.

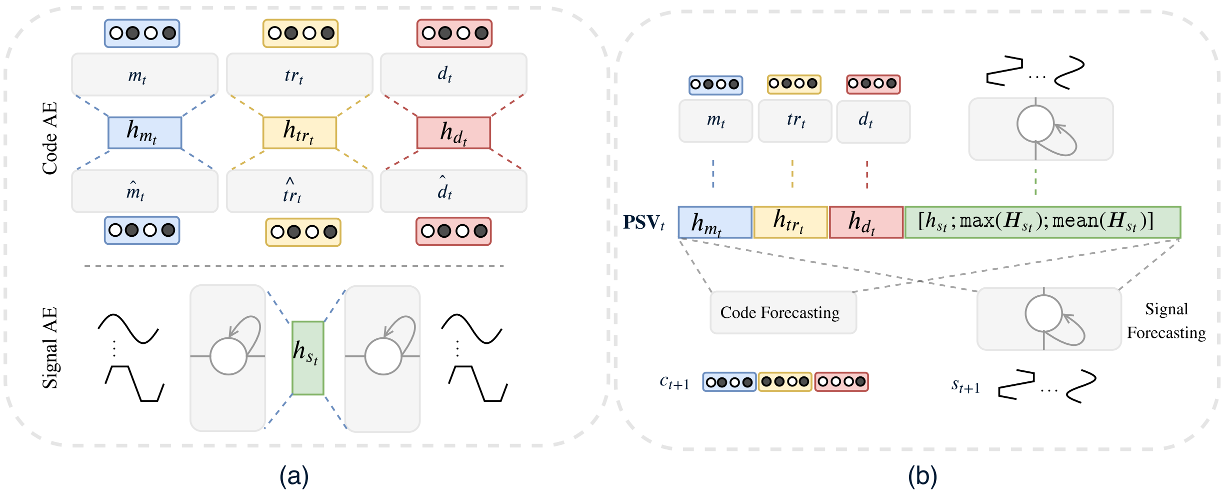


Figure 3.2: Overview of the proposed method: (a) autoencoding step for each datastream, data fed at this stage is predicting data occuring in the same stay. (b) forecasting step using the concatenation of the embeddings ($\mathbf{PSV}_t$). The next time step is predicted in this step.

3.3 Proposed Method

In the current presentation, we are interested in learning a patient embedding using both clinical time-series signals and concepts. The overview of our method is a two-step unsupervised task (Figure. 5.1): the first task is an auto-encoding task in which networks are forced to compress the representation, and the second task is a unified forecasting task using the concatenation of the learned intermediate representations incorporating surrounding context into the representation.

To introduce the problem setting, consider a sequence of EHR data that consists of finite number of medical concepts $\mathcal{C} = \mathcal{M} \cup \mathcal{D} \cup \mathcal{T}$, where $\mathcal{M}$ is the set of medication codes, $\mathcal{D}$ is the set of diagnosis codes, and $\mathcal{T}$ is the set of treatment codes. Clinical codes are typically treated as sets, in which order is not of significance. We simply concatenate the codes in the order in which they are recorded in a patient's record. Additionally, a set of clinical time-series signals $\mathcal{S}$ are measured at varying frequencies, where different patients could have different sets of signals.

The EHR data record of a patient may contain multiple hospital admissions. The admissions could each contain multiple visits to an ICU. As will be explained in section 3.3.2, the dataset we apply the method on does not provide enough information to distinguish the order of hospital visits for a patient. As a result, we treat each admission independent of past hospital admissions and hence a patients data and timestamps are limited to the hospital admission. With restriction in mind, then the longitudinal patient data for a hospital visit can be written as $D_n = \{\mathcal{I}_1, \mathcal{I}_2 \cdots \mathcal{I}_n\}$, where each $\mathcal{I}_i$ denotes an ICU visit with varying duration. An ICU visit contains the data sequence $\mathcal{I}_i^T = \{(m_1, d_1, tr_1, s_1), (m_2, d_2, tr_2, s_2) \cdots (m_T, d_T, tr_T, s_T)\}$, where formally time units in an ICU visit can be defined as minutes, hours or days.

We will go through the training steps to obtain patient representation during hospital admission.

3.3.1 Unsupervised Autoencoder

Briefly, an Autoencoder is a neural network that learns to reconstruct its original input with the goal of learning a useful representation. The Autoencoder model contains an encoding block followed by a decoder. The encoding maps the input $x \in \Re^d$ to an intermediate representation $\Re^m$. Typically m is chosen to be less than d, and is called a bottleneck mechanism forcing the network to compress the high dimensional input to a smaller dimension. The decoder is then tasked to map the representation back to the original data-space $\Re^d$.

If we let f_θ and g_ϕ denote the encoder and decoder block respectively, where θ and ϕ are the model parameters, then the objective can be written as follows:

$$\boldsymbol{e} = f_\theta(x), \hat{x} = g_\phi(e)$$

$$\mathcal{L}(x, \hat{x}) = \frac{1}{N} \sum_{i}^{N} ||x - \hat{x}||_2^2$$

The MSE loss is used for real-valued reconstruction, whereas for multiclass one-hot vectors, the cross-entropy loss is better suited. Ideally, a good performing autoencoder given sufficient context should reconstruct the input with minimal distortion.

3.3.1.1 Medical Concept Embedding

Clinical concepts are high dimensional and sparse. To embed this data input, we use an autoencoder, which takes as input clinical concepts occurring throughout a patient's ICU visit. As medical concepts are time stamped in an ICU visit, it is important to take into account the sequential dependencies as well. To this end, we use multiple transformer networks layers to learn the embeddings [14]. Transformers are known to be suited for sparse sequential data and learning inherent structures present in the input by using the self-attention mechanism. Each concept type is fed to its network with an autoencoder loss (Figure. 5.1a). As we consider three code types, three transformer networks are trained independently on the respective code inputs. More concretely the diagnosis network is given a sequence of diagnosis codes $\{d_1, d_2 \cdots d_T\}$ the codes are first embedded using an embedding matrix $W \in \Re^{|\mathcal{D}| \times d}$, where the embeddings are obtained as $e_t = W^T d_t$. The set of embeddings $\{e_1, e_2, \cdots, e_T\}$ are then fed to transformer decoder layers. A triangular mask is applied so that the output of the transformer at time t can only attend to previous time steps $t-1, t-2, \cdots$ to maintain causality. The objective of the network is then to predict the input following the AE bottleneck

$$L(d_t, \hat{d}_t) = -\sum_i d_i \log(\hat{d}_i) + (1 - d_i) \log(1 - \hat{d}_i)$$

Where $\hat{d}_i$ is the predicted set of diagnosis codes, and d_i is the input multi-hot vector.

3.3.1.2 Clinical Time Series Embedding

Seq2Seq models are well suited for scenarios that require mapping an input sequence to an output sequence. While Seq2Seq models are commonly used for supervised tasks and language modeling, we use it to embed the clinical time series of a patient into a representation by minimizing the reconstruction error between the input and output. Similar to autoencoders, Seq2Seq models have both an encoder and a decoder block. We use Gated recurrent units (GRU) cells, a recurrent neural network variant, which is known to benefit from being able to learn faster and handle longer sequences, as the basic building block of the encoder and decoder of the Seq2Seq model. The input to this network is a sequence of time-series that accompanies the patient's clinical concepts at the same corresponding time denoted as $\{s_1, s_2 \cdots s_T\}$, where each sample s_t is a feature vector $\Re^{|S|}$. As noted earlier, patients may have missing attributes at each time step. To this effect, we present a mask vector $m_t \in \{0,1\}^{|S|}$, where a 1 represents whether the feature has been observed or not. The Seq2Seq model is then trained by windowing the clinical time series, defined at a particular granularity of time, with window size w and setting the objective to the MSE autoencoder loss. We slightly modified the objective to penalize predictions solely when there is an observed value at the corresponding same time step. Formally written as

$$\mathcal{L}(S_{1:T}, \hat{S}_{1:T}) = \frac{1}{N} \sum_{k}^{\frac{T}{w}} m_{wk:w(k+1)} \cdot ||s_{wk:w(k+1)} - \hat{s}_{wk:w(k+1)}||_2^2$$

where $m_{wk:w(k+1)}$ is the mask ranging from time wk to $w(k+1)$, similarly $\hat{s}_{wk:w(k+1)}$ is the output of the seq2seq model and $s_{wk:w(k+1)}$ is the input signal.

3.3.2 Unsupervised Forecasting Task

In the first step, the data streams were fed into their respective models independently. Next, we want to incorporate in the encoder blocks with knowledge of other representations obtained from other data streams. We do this by combining the representations following the first pretraining phase and further fine-tuning on a forecasting task, that is, predicting the next visits diagnosis, medication, treatment codes, and vitals (Figure.5.1 b).

To combine the learned diagnosis, medication, treatment and clinical time series we simply concatenate the representations to create a $\text{PSV}_t = [e_{d_t}; e_{m_t}; e_{tr_t}; e_{s_t}]$ at time t, which we call the patient status vector (PSV).

Each code representation $\{e_{d_t}, e_{m_t}, e_{tr_t}\}$ is obtained by embedding the set of codes until time t and taking the final hidden representation of the transformer network as the representation of the code. For example, the set of diagnosis codes $\{d_1, \cdots d_t\}$ are embedded using the transformer network creating a set of representations $H_d = \{h_{d_1} \cdots h_{d_t}\}$. Then the final hidden representation $e_{d_t} = h_{d_t}$ is used.

On the other hand as there are many more data points for clinical time series, the accumulated vitals until time t are first windowed using the same window size w in the autoencoding step. The resulting is a set of sequences $\{s_{1:w}, s_{w:2w}, \cdots s_{w(\frac{t}{w}-1):t}\}$. Subsequently, a hidden representation is obtained from the encoder cell in the Seq2Seq model $H_s = \{h_{s_1}, \cdots, h_{s_t}\}$ and the representation for the signal portion is obtained as

$$e_{s_t} = [h_{s_t}; \texttt{maxpool}(H_s); \texttt{meanpool}(H_s)]$$

where the components are concatenated, this is done as the time-series signals could get very lengthy and information may get lost if we were only to consider the last representation of the time series. Intuitively, by taking the max-pool, we are looking at a part of the signal resulting in high activation, the mean-pool a baseline measure, and the final representation as to the most recent condition of the patient's signal.

Additionally, we add the patients demographics $z_t \in \Re^{|Z|}$ containing, age, weight, height, and gender where discrete values are coded as one hot vectors. The final representation is obtain as

$$\text{PSV}_t = [e_{d_t}; e_{m_t}; e_{tr_t}; e_{s_t}; z_t]$$

This representation is used for the forecasting task, which has two components: 1) predicting the set of codes at the next time step and 2) predicting the set of vitals for the next time window. This objective of the unsupervised forecasting step can be written as

$$\frac{1}{2T} \sum_{t=1}^{T} \sum_{0 \leq j \leq w, j \neq 0} \log(p(c_{t+j}|\text{psv}_t)) +$$
$$\frac{1}{2\frac{T}{w}} \sum_{i}^{\frac{T}{w}} m_{wk:w(k+1)} \cdot ||s_{wk:w(k+1)} - \hat{s}_{wk:w(k+1)}||_2^2$$

where a fully connected layer followed by softmax is used to model $p(c_{t+j}|\text{PSV}_t)$. On the other hand, $\hat{s_t}$ is a GRU decoder model with hidden input as the concatenation of medical code representations along with the generated encoder representations. Intuitively, this pretraining step will force the networks to learn the surrounding context of other data streams, which could help in predicting a patient's status.

Based on the suggested representation learning scheme, the trained networks from the autoencoding and forecasting steps are used to embed patients given a medical record. This embedding is fed to a simple multi-layer perceptron classifier for downstream tasks. Training details and hyperparameters used for each specific task are detailed in the experiment section.

3.4 Experiments

3.4.1 Data

The dataset used is the publicly available eICU collaborative Research Database v2.0 [52]. The eICU consists of over 200,000 Intensive Care Unit (ICU) records collected from over 250 hospital sites in the United States, between 2014 and 2015. The data is indexed through unique patient identifiers, where each patient could have multiple hospital admissions, and within each hospital admission, they could be admitted to an ICU multiple times. Timestamps in a patient's record are referenced from the ICU admission stay, in which patients could have a certain delay before being admitted to an ICU. As the tables provided in the dataset do not allow one to order hospital admissions, the downstream tasks are limited to within hospital admissions, and further, the patients' history is limited to ICU visit. From the encounter records, we extract diagnosis codes, medication codes, and treatment codes. Additionally, we extract periodic vital signals collected regularly by bedside monitors.

3.4.1.1 Preprocessing

Encounters in the ICU vary significantly from patient to patient. As a result, we preprocess the dataset to include sufficient examples for learning. Patients younger than 16 years of age are removed. The dataset contains patients with burns and organ donors or admission after a transplant which are also removed from the dataset. The periodic bedside monitoring devices are down-sampled to median values every 5 minutes by the original creators of the dataset; We further down-sample by binning the values every 60 minutes (12 samples). The median is used for each bin. Following the preprocessing steps, we normalize signals and patient demographics to have zero mean and unit variance. We summarize the statistics of the compiled cohort in Table 3.1. Two cohorts are considered of different lengths: short ICU visits of 1 hour to 24 hours, and extended ICU visits of 24 hours to 720 hours duration. Our preprocessing steps closely resemble that of APACHE IV [53], a gold-standard metric in which we have used similar exclusion criteria.

Table 3.1: Cohort Statistics of Compiled Data

	1h-24h	24h-720h
# hospital visits	44190	95649
# ICU stays	46664	110270
# of diagnosis codes (avg/visit)	918 (1.17)	918 (3.075)
# of medication codes (avg/visit)	1412 (8.52)	1412 (21.66)
# of treatment codes (avg/visit)	2711 (1.12)	2711 (2.85)
length of signals (avg, min, max)	(14.6, 0, 205)	(87, 0, 832)
# of in-hospital mortality	3249	6157
# of within visit ICU readmissions	10858	21742

3.4.2 Downstream tasks

To evaluate the usefulness of the learned representation, we train a classifier on top of the representation on downstream tasks and evaluate its generalization performance.

Mortality: Given a patient's record and patients history, predict the patient's death during the ICU stay. This is a binary prediction task.

Readmission Similarly, in this task, we focus on predicting whether the patient will be readmitted to ICU again within the same visit.

3.4.3 Training Details & Evaluation & Baselines

The unsupervised training follows the method described in Section 3.2.

The medical concept autoencoder blocks are transformer layers with multi-head attention, as described in [14]. This block contains multiple hyperparameters namely number of multi-head attention heads n_{head}, dimension for each head d_{head} and the final model representations d_{code}. We set these to 8, 64, and 256 respectively for all three medical concept embedding blocks two such layers are stacked. The network is trained using Adam [34] with a cosine annealing schedule and a period of 50 epochs. The initial learning rate is set to 0.00025.

The Seq2Seq model is trained by setting the window size for clinical signals to 24 (i.e., a full day). Missing values are imputed with the median of the statistics of the complete dataset if the patient does not have any history for the vital; otherwise, the patient's average is used to impute the corresponding missing signal. The encoder and decoder blocks are bidirectional GRU cells where the encoder hidden layer dimension is set to $d_{enc} = 128$, and the decoder is set to 256. A step learning rate with initial lr set to 0.001, and step decay rate of 50 is used to train the network for 100 epochs.

Following the autoencoding training, the dataset is filtered to contain visits with multiple ICU stays for the forecasting step. To avoid catastrophic forgetting, we gradually unfreeze the blocks and allow them to train for 2 epochs [54]. This is similar to chain thaw proposed in [55], where each layer is trained at a time. Though here, we gradually unfreeze the whole model and allow it to learn altogether. Our final patient status vector representation is of dimension 1762 fed to a two-layer fully connected layer with ReLU as activation and dropout set to 0.1 in between layers.

We implemented all the models with Pytorch 1.0 [37]. For training the models, we use the Adam optimizer [34]. In all experiments, the batch size is set to 64.

3.4.3.1 Evaluation

For all experiments, we use 15% of the data selected randomly as the test set. The remainder is used for training/validation splits. In all experiments, the presented values are the average of 5 experiments for each task.

Area under the precision-recall (AU-PR): this metric is the cumulative area under the curve by plotting precision and recall while varying the outputs $P(y = 1|(c_1 : t, s_1 : t))$ true/false threshold from 0 to 1.
Receiver operating characteristic curve (AU-ROC): This metric is the area under the plot of the true positive rate against false positive rate while varying outputs $P(y = 1|(c_1 : t, s_1 : t))$ true/false threshold from 0 to 1.

3.4.3.2 Baseline Models

- **Transformer Embedding**: The set of medical concepts in the records are embedded by training a transformer network using the skip-gram model. This representation is used as a patient representation for downstream tasks.
- **Seq2Seq** (Unsupervised): A Seq2Seq model is trained on clinical times series portion of the data, which can then be used to embed patients signals for the downstream task.
- **Seq2Seq** (Semi-supervised): A Seq2Seq model is trained on clinical times series portion of the data, and further fine-tuned on downstream tasks.
- **Transformer**$_{code}$ + **Transformer**$_{signal}$ Two transformer networks are fed independent data streams and concatenated for downstream supervised tasks. The method is supervised.
- **Ours - Supervised**: The proposed model with the same complexity is trained with the raw data and directly trained on downstream tasks.

We also perform an ablation on different components of the patient status vector by including/excluding portions of the representations

3.4.4 Limited Data Setting

To show the usefulness of unsupervised pretraining, we run our pretraining method in a limited data setting. That is, the model is pretrained on a small subset of the dataset and evaluated on the rest. We run experiments on random train splits sizes of [0.05, 0.15, 0.25, 0.5, 0.75, 0.9]%. Our model is then compared to the proposed model with the same complexity in terms of network size and trained with supervised signal given raw input. Ideally, the model should outperform the supervised method when there is limited data as pretraining allows the network to generalize better by avoiding bad local minimas, which the supervised method would fit into.

3.5 Results

To evaluate the PSV representation, we consider both a short duration cohort and a long duration cohort. This is done as the dynamics of short term ICU stays, and long term ICU stays vary greatly. The results for both downstream tasks are summarized in Table 3.2&3.3.

From the tables, the ablation of the different components for the PSV model is done by first pretraining the model in an unsupervised fashion and freezing the pre-trained network on downstream tasks. Empirically, from both tables, it is evident that the network benefits from both code and signal representations on the defined tasks.

Two of the presented baseline models use only portions of the complete data under consideration, and to make a fair comparison, we can compare them with the respective components of the PSV model accordingly. For example, Seq2Seq [51] can be compared to PSV `(Signal)`. Similarly, Transformer [39] can be compared to PSV `(Code)`. In both cases, the difference is largely in the unsupervised training step. As a result, by leveraging both single visit ICU data and multi-visit ICU patients, we can achieve a better initialization for downstream tasks leading to improved predictive performance.

We also compare our method with baselines that make use of the same inputs of our complete PSV model. Referring to both tables, both supervised methods which have comparable sizes to our model converge to lower accuracies. This makes sense as unsupervised pretraining allows the network to learn model distribution, whereas supervised methods tasked with directly learning the input to output mapping, making it harder to optimize the model's parameters.

Lastly, from the results, the prediction performance drops (significantly in some cases) when comparing short visits and long visits. This could largely be due to dynamics that are present in longer ICU visits or human factors that are not present/captured in the dataset.

3.5.1 Calibration

The reliability of a model's confidence is critical in health settings and calibration plots are a common measure for the model's reliability [56, 57]. A method for learning patient representation should lend itself to well calibrated model, which intuitively, means a model predicting with 80% confidence should be correct 80% of the time. Generally, more recent methods are found to not be well calibrated [56]. To this end, we provide calibration plots of our model for both readmission and mortality tasks in Fig. 3.3. The plots are obtained by binning the models predictions and evaluating the models accuracy at each bin level. From this figure, we can see that the model accuracy increases as its confidence in prediction also increases. The readmission task follows the ideal scenario more closely compared to the mortality task. Note that in a binary classification problem for any classifier the baseline confidence is 0.5.

3.5.2 Visualization

We illustrate the t-SNE plot for our learned representation for both readmission and mortality tasks by reducing the patient status vector across all sample points to its 2 largest principal components. Referring to Fig 3.4 a), the readmission task is relatively better

[1]Results reported using our re-implementation, further the input are limited to the same set of vitals/codes under consideration.

Table 3.2: Mortality downstream task ablation, and comparison with baselines. Results reported are average of 5 runs on a test split of 15% and the std are reported in parenthesis.

Dataset (`Mortality task`)	1h-24h		24h-720h	
	PR-AUC	ROC	PR-AUC	ROC
PSV (`Code+Signal`)	62.46 (± 0.20)	90.06 (± 0.12)	48.88 (± 0.13)	85.90 (± 0.09)
PSV (`Code`)	45.35 (± 0.22)	81.69 (± 0.29)	30.50 (± 0.22)	77.98 (± 0.14)
PSV (`Signal`)	49.42 (± 0.18)	82.27 (± 0.04)	31.32 (±0.10)	82.27 (± 0.04)
PSV (`Semi-supervised`)	65.40 (± 0.35)	89.10 (± 0.59)	53.76 (± 0.26)	85.15 (± 0.66)
Seq2Seq [51][1]	7.73 (± 0.60)	51.32 (± 0.31)	8.17 (± 0.05)	61.90 (± 0.19)
Seq2Seq (`Semi-supervised`)	8.58(±0.32)	51.23(± 0.22)	19.22 (± 0.06)	61.63 (± 61.63)
Transformer [39][2]	11.18 (± 0.35)	53.01 (± 0.64)	10.67 (± 0.30)	50.45 (± 0.88)
Transformer (`Code`) + (`Signal`)	58.12 (± 0.46)	78.28 (± 0.39)	47.71 (± 0.35)	83.45 (± 0.29)
PSV (`Supervised`)	64.61 (± 0.32)	88.38 (± 0.13)	51.24 (± 0.31)	81.25 (± 0.26)

clustered compared to the mortality task.

3.5.3 Limited Data Setting

Finally, we also show the usefulness of unsupervised pre-training in limited data settings. Referring to Fig 3.5, our semi-supervised model consistently outperforms its supervised counterpart, more so in the lower end.

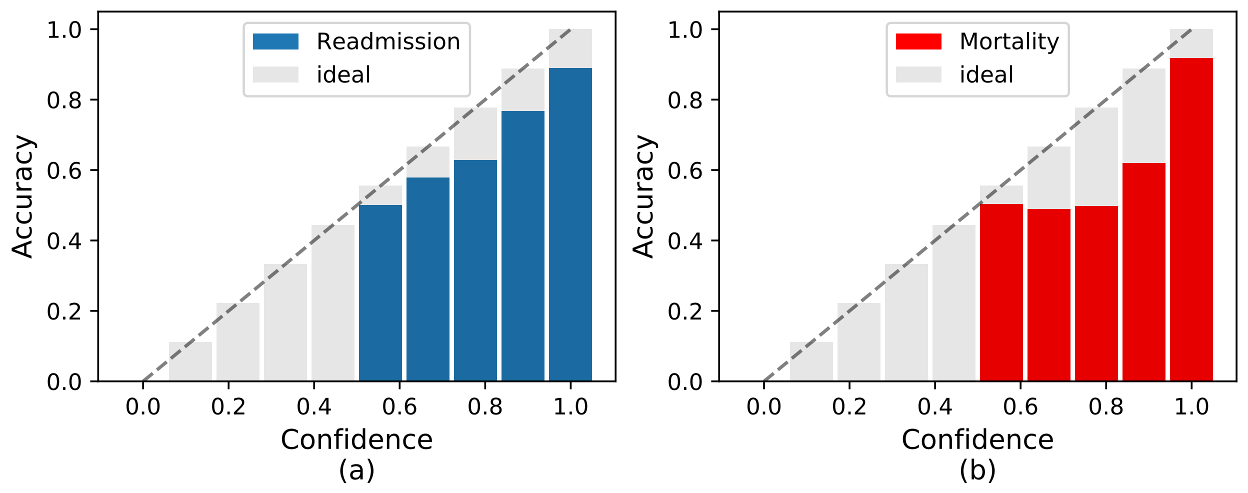


Figure 3.3: Calibration plots for both readmission (a) and mortality (b) tasks. The ideal calibration plot is plotted in grey for both Figures.

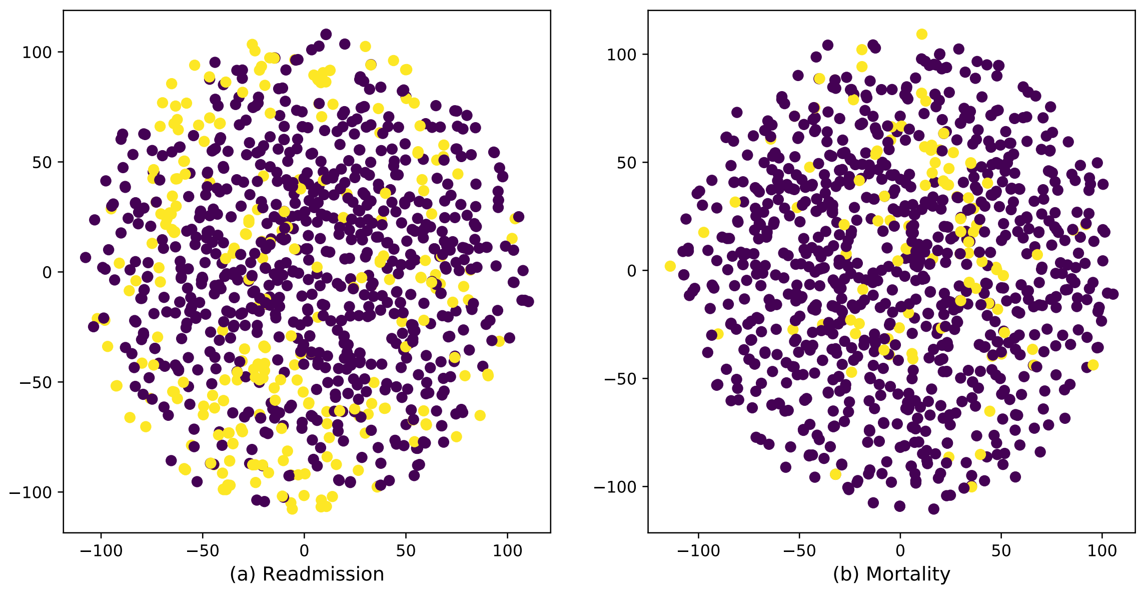


Figure 3.4: t-SNE visualization for different subset of learned representations for both readmission and mortality task. From the Figures in the case of a) the positive targets mainly surround the border whereas in the mortality case b) they are clustered more so in the center.

Table 3.3: Readmission downstream task ablation, and comparison with baselines. Results reported are average of 5 runs on a test split of 15% and the std are reported in parenthesis.

Dataset (Readmission task)	1h-24h		24h-720h	
	PR-AUC	ROC	PR-AUC	ROC
PSV (Code+Signal)	61.25 (± 0.26)	80.99 (± 0.11)	57.86 (± 0.19)	80.94 (± 0.09)
PSV (Code)	57.24 (± 0.56)	79.58 (± 0.12)	49.63 (± 0.38)	76.15 (± 0.24)
PSV (Signal)	30.47 (± 0.13)	59.35 (± 0.15)	30.34 (± 0.10)	59.22 (± 0.14)
PSV (Semi-supervised)	69.02 (± 0.42)	83.40 (± 0.23)	68.04 (± 0.51)	82.25 (± 0.24)
Seq2Seq [51][2]	26.43 (± 0.56)	51.45 (± 1.13)	20.30 (± 0.18)	52.35 (± 0.34)
Seq2Seq (Semi-supervised)	26.68(± 0.19)	51.80 (±0.54)	22.31 (± 0.11)	52.18 (± 0.42)
Transformer [39][2]	28.22 (± 0.60)	58.82 (± 0.43)	27.70 (± 0.79)	59.45 (± 0.91)
Transformer (Code) + (Signal)	56.23 (± 0.34)	75.98 (± 0.53)	54.79 (± 0.49)	76.45 (± 0.36)
PSV (Supervised)	60.12 (± 0.33)	80.73 (± 0.43)	58.41 (± 0.21)	78.79 (± 0.44)

3.6 Conclusion

In this paper, we have introduced an unsupervised patient status vector embedding scheme for EHR patient longitudinal data. The method effectively leverages both single-visit ICU patients and multi-visit ICU patients using a two-step autoencoding step. We have evaluated the proposed method using two cohorts compiled from the eICU EHR dataset of different duration by using periodic vital signals and medical codes as input to our model. From empirical results on downstream tasks, the proposed unsupervised learning approach out-

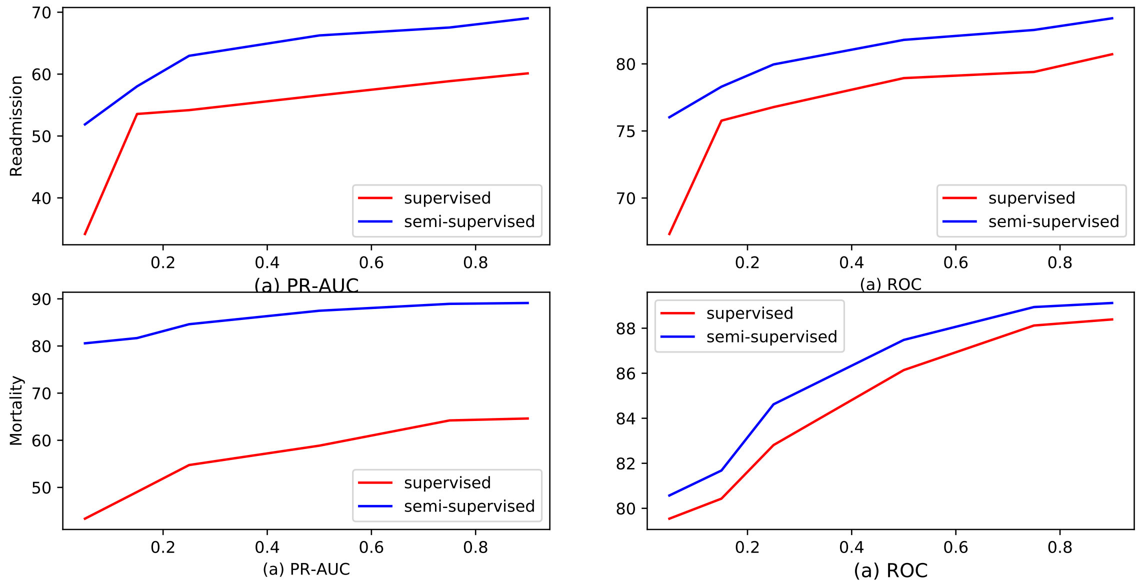


Figure 3.5: Comparison of supervised and unsupervised PSV model trained on limited data.

performs previous work. Lastly, presented in the results, long-stay ICU visit patients present a bigger challenge for modeling; as EHR data do not necessarily contain human factors that could play in the prognosis of a patient, future work could leverage different modes of data, which were not considered in this work.

CHAPTER 4

Contrastive Mixup: Semi-Supervised learning for Tabular Domain

Recent literature in self-supervised has demonstrated significant progress in closing the gap between supervised and unsupervised methods in the image and text domains. These methods rely on domain-specific augmentations that are not directly amenable to the tabular domain. Instead, we introduce Contrastive Mixup, a semi-supervised learning framework for tabular data and demonstrate its effectiveness in limited annotated data settings. Our proposed method leverages Mixup-based augmentation under the manifold assumption by mapping samples to a low dimensional latent space and encourage interpolated samples to have high a similarity within the same labeled class. Unlabeled samples are additionally employed via a transductive label propagation method to further enrich the set of similar and dissimilar pairs that can be used in the contrastive loss term. We demonstrate the effectiveness of the proposed framework on public tabular datasets and real-world clinical datasets.

4.1 Introduction

Deep learning has shown tremendous success in domains where large annotated datasets are readily available such as vision, text, speech via supervised learning. Implicitly learned by these models is an intermediate representation that lends itself useful for downstream tasks. Unfortunately, in many settings such as healthcare, large annotated datasets are not readily available to enable learning such valuable representations. As a result, there has been a push towards learning these in an unsupervised or semi-supervised manner as unannotated data on the other hand may be readily available for free and a lot of it in many cases. Recent literature has shown significant progress towards learning these useful representations without human-annotated data, closing the gap between supervised and unsupervised learning, and in some cases demonstrating superior transfer learning properties compared to its supervised counterpart [58, 59].

Self-supervised methods have emerged as a promising approach to achieving appealing results in various applications without requiring labeled examples. This is typically done via pretext tasks closely related to the downstream tasks of interest and typically differs from domain to domain. For example, in the image domain colorization [60], jigsaw puzzle[61], rotation prediction [62] have been previously presented as pretext tasks useful for learning such representations. Similarly, in the text domain, commonly used pretext tasks, such as predicting masked words and context words, have been widely used [63, 5]. More recently, contrastive learning methods introduced leverage domain specific transformations to create multiple semantically similar examples such as random cropping or flipping for images that preserve the semantic meaning and encourage the network to be invariant to such transformations achieving great success. Such pretext tasks and transformations cannot be readily applied that do not have the same structural information, as an example tabular data[1].

It is not clear how to generate new semantically similar examples for tabular data. Moreover, in many settings, tabular data contains both categorical and continuous features which require different treatment. In this work, we focus on tabular data settings that contain a small set of annotated samples and a relatively sizeable unlabeled set of samples. Specifically, we propose a framework for improving downstream task performance in this semi-supervised setting. Our method consists of a semi-self-supervised pretraining step where a feature reconstruction pretext task and a supervised contrastive loss term are used. Various forms of Mixup augmentation [64] has been used in the image domain, where new examples are created by taking convex combinations of pairs of examples. This may lead to low probable samples in the dataspace for tabular data. Instead, we leverage the manifold assumption [2] and mix samples in the latent space to create multiple views for our contrastive loss term. The unlabeled subset is further leveraged by pre-training the encoder and using label prop-

[1]Tabular data contains a set of rows (examples) and columns (features) that may be permutation invariant.

[2]*Manifold Assumption*: High-dimensional data lies (roughly) on a low-dimensional manifold.

agation [65] to generate pseudo-labels for the unlabeled samples. Subsequently, the trained encoder and samples, for which we have generated pseudo-labels for, are transferred to a downstream task where a simple predictor with Mixup [64] augmentation is trained. We show that our proposed framework leads to improvements on various tabular datasets, such as UCI and Genomics (UK Biobank).

4.2 Related Works

Our work fits the semi-supervised learning framework [?] where both labeled and unlabeled samples are used to improve downstream task performance. We draw from recent literature in self-supervised representation learning, pseudo-labeling [66] and Mixup based supervised learning [64].

At the core of the self-supervised methods are pre-text tasks, where labels are created from the raw unlabeled data itself, and supervised losses are then used to learn useful representations for downstream prediction tasks. In these lines of works, examples of domain-specific pre-text tasks such as jigsaw puzzle [67], colorization [60], relative positioning prediction [68] have been introduced for images, masked word prediction [63, 69], next sentence prediction [70] for text. There is also existing work on self-supervised/semi-supervised learning methods. For example, a similar in-painting task [71] can be used to predict masked features in a row as done in [72, 73]. On the other hand, many recent self-supervised methods are based on contrastive representation learning [59], in which domain-specific augmentation (e.g., random crop, random color distortion for images) are used to create "similar" samples, and the normalized cross-entropy loss [74, 75] is used to increase the similarity of "positive" pairs in the latent space, and decrease the similarity of "negative" pairs. A downside of generating negative and positives without label information is that examples belonging to the same class may be pushed apart. In [76] authors leverage label information to consider many "similar" examples to be pulled closer to one another and farther away from the dissimilar examples. As these methods leverage properties inherent to the raw data, they are not amenable to the tabular domain, which is the focus of this work.

The setting we are considering fits the semi-supervised learning framework. Prior work on semi-supervised learning can be broadly separated into two main categories: methods that add an unsupervised loss term to the supervised task as a regularizer, e.g., [61, 77, 78] and methods that assign pseudo labels [66] to the unlabeled samples. Recently, [79] proposed VIME, a state-of-the-art semi-supervised method for the tabular domain where they leverage consistency regularization and in-painting [71] inspired augmentation. In [66] the current network trained on labeled samples is used to infer pseudo-labels on unlabeled samples using a confidence threshold, which is then treated similar to labeled samples to minimize entropy. Transductive learning is more generic in that instead of training a generic classifier, the goal is to used patterns in the labeled set to infer labels for the unlabeled set. Label propagation has been widely used in transductive learning in the image domain in an online fashion where CNN features are used for few-shot learning [80]. Along this line of work, recently [65] use label propagation in an offline fashion by treating the labeled and unlabeled samples as a bipartite graph where edges computed via diffusion similarity [81]. In this work, we propose a semi-supervised framework for the tabular domain where we leverage Mixup [64] based augmentation, which interpolates samples using a convex combination and assigns soft labels according to the mixing ratio in the latent space [82] and encourage samples interpolated from the same class to have high similarity.

To present our method we formulate the self-supervised and semi-supervised problem. Consider a dataset with N examples: Our assumption is that there is a small subset of this dataset for which labels are available: $\mathcal{D}_L = \{(x_i, y_i)\}_{i=1}^{N_L}$, and the rest of the dataset is unlabeled: $\mathcal{D}_U = \{(x_i)\}_{i=1}^{N_U}$ where x_i are observations sampled from a data-generating distribution $p(x)$ and $y_i \in \{0, 1, \cdots, c\}$ is a discrete label set. We consider settings where the majority of the data is unlabeled i.e. $|\mathcal{D}_L| \ll |\mathcal{D}_U|$. In supervised learning a classifier $f : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ is a function learned by an ML algorithm which aims at optimizing f for a given loss function $l_A(\cdot)$ i.e. $f = \min_{f \in \mathcal{F}} \sum_{i=1}^{N} l_A(f(x_i), y_i)$. In this limited labeled data regime a supervised model is most likely to overfit, hence we propose to use the unlabeled samples to improve the models generalization.

4.2.1 Self-Supervised Learning

Self-supervised methods leverage unlabeled data to learn useful representations for downstream prediction tasks. Many techniques have been proposed for images where useful visual representations are learned through pre-text tasks such as in-painting, rotation, jig-saw [67, 71, 62], and more recently, the gap between supervised and unsupervised models have drastically been reduced through contrastive representation learning method [58, 59]. Generally, in contrastive representation, learning a batch of N samples is augmented through an augmentation function Aug(.) to create a multi-viewed batch with $2N$ pairs, $\{\tilde{x_i}, \tilde{y_i}\}_{i=1\cdots 2N}$ where $\tilde{x}_{2k}$ and $\tilde{x}_{2k-1}$ are two random augmentations of the same sample x_k for $k = \{1, \cdots, N\}$. The samples are fed to an encoder $e : x \rightarrow z$ which takes a sample $x \in \mathcal{X}$, to obtain a latent representation $z = e(x)$. Typically when defining a pre-text task, a predictive model is trained jointly to minimize a self-supervised loss function l_{ss}.

$$\min_{e,h} \mathbb{E}_{(x,\tilde{y})\sim P(X,\tilde{Y})} \left[l(\tilde{y}, h \circ e(x) \right] \tag{4.1}$$

where h maps z to an embedding space $h : z \rightarrow v$. Within a mutliviewed batch, $i \in \mathcal{I} = \{1, \cdots 2N\}$ the self supervised loss is defined as

$$l = \sum_{i\in\mathcal{I}} -\log\Big(\frac{\exp(\mathrm{sim}(v_i, v_{j(i)})/\tau)}{\sum_{n\in\mathcal{I}\setminus\{i\}} \exp(\mathrm{sim}(v_i, v_n)/\tau)}\Big) \tag{4.2}$$

Here, $\mathrm{sim}(\cdot,\cdot) \in \Re^+$ is a similarity function (e.g. dot product or cosine similarity), $\tau \in \Re^+$ is a scalar temperature parameter, i is the *anchor*, $\mathcal{A}(i)$ is the *positive(s)* and $\mathcal{I}\setminus\{i\}$ are the *negatives*. The positive and negative samples refer to samples that are semantically similar and dissimilar respectively. Intuitively, the objective of this function is to bring the positives and the anchor closer in the embedding space v than the anchor and the negatives, i.e. $\mathrm{sim}(v^a, v^+) > \mathrm{sim}(v^a, v^-)$, where v^a is the anchor and v^+, v^- are the positive and negative respectively.

4.2.2 Semi-Supervised Learning

In semi-supervised learning (SSL), the dataset is comprised of two disjoint sets D_L. D_U, where predictive model f is optimized to minimize a supervised loss, jointly with an unsupervised loss. In other words:

$$\min_{f} \mathbb{E}_{(x,y)\sim P(X,Y)} \big[l(y, f(x))\big] + \beta \mathbb{E}_{(x,y_{ps})\sim P(X,Y_{ps})} \big[l_u(y_{ps}, f(x))\big] \tag{4.3}$$

The first term is estimated over the small labeled subset $\mathcal{D}_U$, and the second unsupervised loss is estimated over the more significant unlabeled subset. The unsupervised loss function l_u is defined to help the downstream prediction task, such as consistency loss training [67, 77], or in our case, a supervised objective on pseudo-labeled samples [66].

4.3 Method

This section describes our proposed method Contrative Mixup, a semi-supervised method for multi-modal tabular data where structural (spatial or sequential) data augmentations are not readily available. To this end, we first propose our semi-supervised training to learn an encoder and subsequently propose to train a classifier using the pre-trained encoder and pseudo-labels.

4.3.1 Semi-Self-Supervised Learning for Tabular Data

We make use of the manifold assumption where high dimensional data roughly lie on a low dimensional manifold and then leverage Mixup [64] based data interpolation for creating positive and negative samples. By doing so we mitigate creating low-probable samples in the original data space.

In our setting we represent the mutli-modal tabular data rows x_i as a concatenation of discrete $D = [D_1, \cdots, D_{|D|}]$ and continuous features $\mathcal{C} = [C_1, \cdots, C_{|\mathcal{C}|}]$. The raw features $x_i \in \Re^d$ are fed through an embedding layer $E : x \rightarrow \bar{x}$ that results in a feature vector $\bar{x} \in \Re^{|C|+\Sigma_i^{|D|} d_{|\mathcal{D}_i|}}$, that is a concatenation of the continuous features $\mathcal{C}$ and embedded discrete features $\mathcal{D}$, where $d_{|D_i|}$ is the embedding dimension for each discrete feature $\mathcal{D}_i$. The

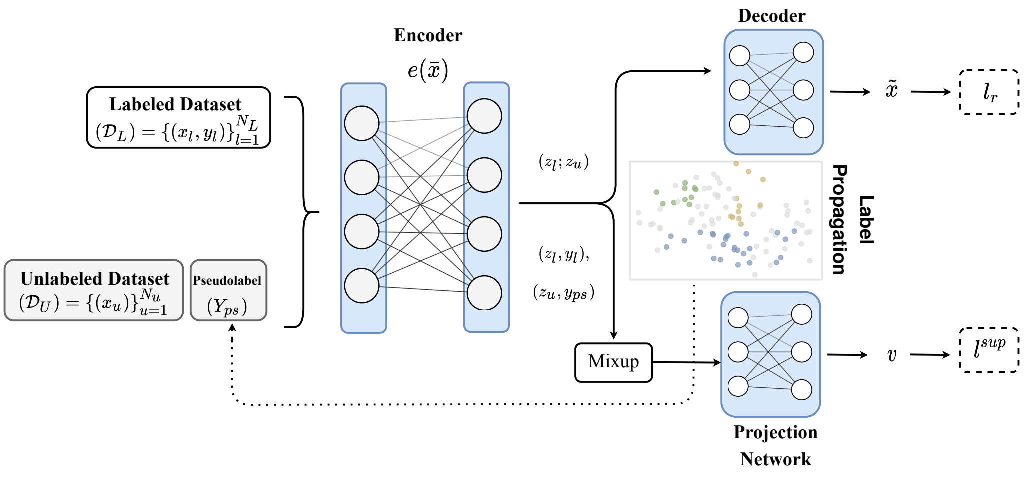


Figure 4.1: Overview of our semi-self-supervised framework. The encoder is trained using both labeled and unlabeled subsets via the reconstruction loss and contrastive loss terms. Pseudo-labeles are used

embedded features are fed to an encoder $z = e(\bar{x})$, and subsequently fed to a feature estimation pre-text task, as well as a semi-supervised contastive loss term shown in Figure 5.1.

In the tabular domain, data augmentation commonly used in the image domain cannot be used. Instead, we propose to interpolate between samples of the same class to create positive examples and use a supervised contrastive loss term in the latent space. Given a batch of labeled examples $\mathcal{D}_{\mathcal{B}} = \{x_k, y_k\}_{k=1}^{K}$, we create a new labeled sample $(\hat{x}, \hat{y})$ by interpolating within the same labeled pair of examples

$$\hat{x} = \lambda x_1 + (1 - \lambda)x_2 \tag{4.4}$$

where λ is a scalar sampled from a random uniform $\lambda \sim \mathcal{U}(0, \alpha)$ with $\alpha \in [0, 0.5]$. The newly generated sample $\hat{x}$ will be λ close to x_1 and $1-\lambda$ to x_2 with the same label as x_1 and x_2, i.e. $y_1 = y_2 = \hat{y}$. As opposed to randomly interpolating between samples and enforcing closeness between samples of different labels, we encourage samples of the same label to lie close to one another in the latent space.

Applying Mixup in the input space for tabular data may lead to low probable samples due to the multi-modality of the data and presence of categorical columns. Instead, we map samples to the hidden space and interpolate there. More concretely, given an encoder e, that is comprised of T layers f_t, for $t \in \{1, \cdots T\}$. The samples are fed through to an intermediate representation h_t at layer t. This layer contains a more abstract representations of the input samples x_1 and x_2. The samples are interpolated within this intermediate layer as

$$\tilde{h}^t_{12} = \lambda h^t_1 + (1 - \lambda) h^t_2 \tag{4.5}$$

where h^t_i is obtained by feeding samples $\bar{x}_i$ through the encoder until layer t. Subsequently, the newly generated samples $\tilde{h}^t_{i'i}$ as well as the original samples h^t_i are fed through the rest of the encoder layers $t, \cdots, T$ to obtain the latent representation z. In this space we distinguish between z_l and z_u, which are the latent representation of labeled and unlabeled samples respectively in. Note that initially we only consider the labeled portion for the contrastive term, i.e. (z_l, y_l) in Figure. 5.1. We define the contrastive loss term to encourage samples created from pairs of the same class to have high similarity. It is common practice to introduce a separate predictive model to map the latent representations to an embedding space via a projection network h^{proj} where the contrastive loss term is defined. We use supervised contrastive loss [76] for the labelled set $\mathcal{D}_L$ as our augmentation views are within a class. It generalizes Eqn. 4.2 to an arbitrary number of positive samples, due to the presence of labels and examples belonging to the same class are encouraged to have high similarity, making the loss term more sample efficient.

$$l^{sup}_{\tau} = \sum_{i \in \mathcal{I}} \frac{-1}{P(i)} \sum_{p \in P(i)} \log\Big(\frac{\exp(\text{sim}(h^{proj}_i, h^{proj}_p)/\tau)}{\sum_{n \in Ne(i)} \exp(\text{sim}(h^{proj}_i, h^{proj}_n)/\tau)} \Big) \tag{4.6}$$

In the above, $P(i) = \{p | p \in \mathcal{A}(i), y_i = \tilde{y}_p\}$ is the set of indices of positives with the same label as example i, $|P(i)|$ is its cardinality, and $Ne(i) = \{n | n \in \mathcal{I}, y_i \neq y_n\}$. This objective function will encourage mixed-uped labeled samples and anchors of the same sample to be close leading to a better cluster-able representation. In addition to the above loss term the encoder is trained to minimize the feature reconstruction loss via a decoder $f_\theta(\cdot)$

$$l_r(x_i) = \frac{|C|}{d} \sum_{c}^{|C|} ||f_\theta \circ e_\phi(x_i)^c - x^c_i||^2_2 + \frac{|D|}{d} \sum_{j}^{|D|} \sum_{o}^{d_{D_j}} \mathbf{1}[x^d_i = o] \log(f_\theta \circ e_\phi(x_i)^o) \tag{4.7}$$

The semi-self supervised objective function can then be written as

$$L = \mathbb{E}_{(x,y) \sim \mathcal{D}_L} \big[l^{sup}_{\tau}(y, f(x)) \big] + \beta \mathbb{E}_{x \sim \mathcal{D}_U \cup \mathcal{D}_L} \big[l_r(x) \big] \tag{4.8}$$

The encoder is trained using this loss term over K epochs, to warm-start the representations in the latent space prior to pseudo-labeling and leveraging the unlabeled samples.

4.3.2 Psuedo-labeling Unlabeled Samples

Thus far, we have only used the labelled set $\mathcal{D}_L$ in the contrastive loss term l_τ^{sup}. To make use of the unlabeled set using $\mathcal{D}_U$ we proposed to use label propagation [65, 81] after K epochs of training with the supervised contrastive loss term L^{sup}. Given the encoder trained on $\mathcal{D}_L$ for K epochs, we map the small labelled set $\mathcal{D}_L$, and a subset of the unlabeled set $S_U \subset \mathcal{D}_U$ to the latent space z and construct an affinity matrix G

$$g_{ij} := \begin{cases} \text{sim}(z_i, z_j) & \text{if } i \neq j \text{ and } z_j \in \text{NN}_k(i) \\ 0 & \text{otherwise} \end{cases} \tag{4.9}$$

where $\text{NN}_k(i)$ is the k nearest neighbor of sample z_i and $\text{sim}(\cdot, \cdot)\Re^+$ is a similarity measure, e.g. $z_i^T z_j$. We then obtain pseudolabels for our unlabeled samples by computing the diffusion matrix C and setting $\tilde{y}_i := \arg\max_j c_{ij}$, where

$$(I - \alpha\mathcal{A})C = Y$$

Similar to [65, 83] we use conjugate method to solve linear equations to obtain C to enable efficient computation of the pseudo-labels. Here $\mathcal{A} = D^{-1/2}WD^{-1/2}$ is the adjacency matrix, $W = G^T + G$ and $D := \text{diag}(W1_n)$ is the degree matrix. Once we've obtained the pseudo-labels for the unlabeled subset S_U, we train the encoder with unlabeled samples treating the generated labels as ground truth

$$L = \mathbb{E}_{(x,y)\sim\mathcal{D}_L}\left[l^{sup}(y, f(x))\right] + \gamma\mathbb{E}_{(x,y_{ps})\sim S_U)}\left[l^{sup}(y_{ps}, f(x))\right] + \beta\mathbb{E}_{x\sim\mathcal{D}_U}\left[l_r(x)\right] \tag{4.10}$$

The pseudo-labels are updated every f epoch of training with the above loss term.

4.3.3 Predictor

Following the semi-supervised pre-training, the encoder is transferred to the downstream task along with the generated pseudo-labels to train the predictor on the downstream task.

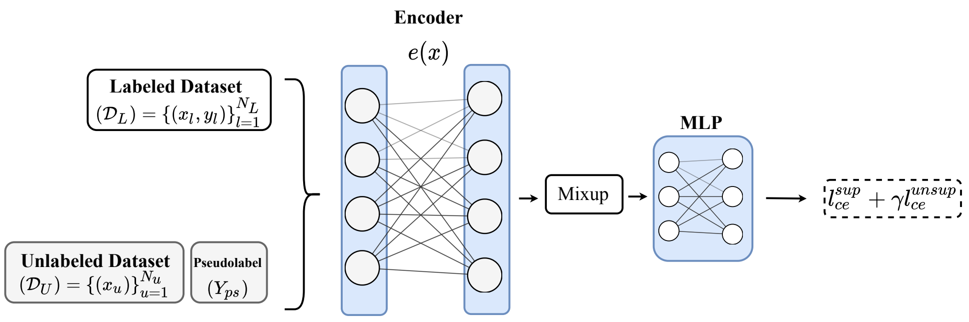

Encoder
$e(x)$
Labeled Dataset
$(\mathcal{D}_L) = \{(x_l, y_l)\}_{l=1}^{N_L}$
Unlabeled Dataset
$(\mathcal{D}_U) = \{(x_u)\}_{u=1}^{N_u}$
Pseudolabel
(Y_{ps})
Mixup
MLP
$l_{ce}^{sup} + \gamma l_{ce}^{unsup}$

Figure 4.2: Overview transfering the semi-supervised pre-training steps to the downstream task. Encoder $e(\bar{x})$ is fixed and the predictor - multilayer perceptron (MLP) is trained using Mixup augmentation. l_{ce}^{x} is the generic cross-entropy loss split into supervised (sup) for labeled subset and unsupervised (unsup) for the unlabeled subset.

We leverage Mixup augmentation [64] in the latent space and feed samples to a set of fully connected layers as depicted in Figure 4.2.

4.4 Experiment & Emperical Results

This section applies the proposed framework on a set of different tabular datasets and application domains to demonstrate its effectiveness. We compare our semi-supervised framework with VIME [79] another semi-supervised approach for the same problem set as a benchmark. To evaluate the pre-training phase, we compare with auto-encoder. We also compare with other semi-supervised method manifold mixup [82]. As a baseline, we include supervised methods, Logistic Regression, a 2-layer multi-layer perceptron network (MLP) that is used as the same architecture amongst other deep methods as the predictor network, and we also include CatBoost [84] as a gradient boosting tree method widely used on tabular data as it supports categorical columns. Additionally, we provide results for including various components of the proposed framework as ablation for the usefulness of each part of the method. In the experiments, self/semi-supervised use the labelled and unlabeled sets $\mathcal{D}_L$, $\mathcal{D}_U$ during training, and supervised models only used the labelled sets $\mathcal{D}_L$. We normalize the continuous columns to 0, 1 using Standard-scaler[3]. The implementation of ContrastiveMixup can be

[3]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

found at https://github.com/¡anonmous¿/ContrastiveMixup

4.4.1 Public Tabular Datasets

We compare the proposed method on four public UCI[4] datasets: MNIST, where examples are interpreted as 784-dimensional feature vectors, UCI Adult, UCI Covertype, more details, are available in the supplementary. We use 10% of the data as labelled and the rest as unlabeled; if the dataset contains an original test set, we use this to evaluate the methods; otherwise, we split the dataset 80% train and 20% test, and the ratios mentioned above follow. As we introduced embedding layers for categorical columns in our method, we choose the best of one-hot encoding categorical columns or embedding layers for other methods. The different variants of our methods for the ablation study are as follows:

- **Supervised**: the pretraining is removed and only the predictor is trained (i.e. MLP)
- **Self-SL only**: the pre-training consisting of labeled contrastive loss term and unsupervised reconstruction loss without pseudolabeling. (i.e. $\lambda = 0$
- **Self-SL + PL**: This is the pre-training with pseudolabeling component added, without mixup component when training the predictor.

[4]https://archive.ics.uci.edu/ml/datasets.php

Table 4.1: Comparison on public tabular datasets.

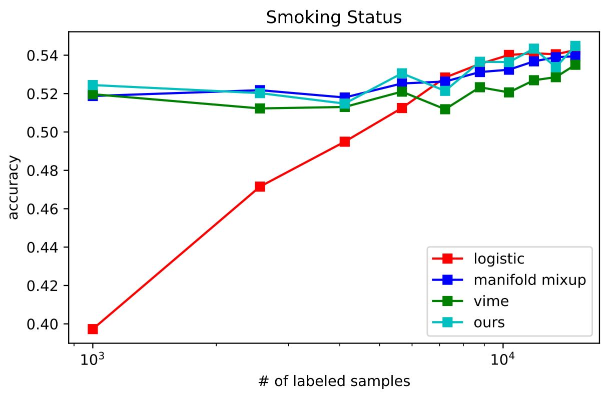
		Dataset			
Type	**Method**	**MNIST**	**Adult**	**Blog Feedback**	**Covertype**
	Logistic	90.12 (±0.0098)	82.41 (±0.0413)	78.91 (±0.022)	70.54 (±0.0087)
Supervised	MLP	93.69 (±0.0234)	83.19 (±0.0663)	79.63 (±0.0519)	75.95 (±0.0202)
	Catboost	(±)	- (±−)	- (±−)	- (±−)
	Catboost (100%)	97.41 (±0.0098)	87.54 (±0.0075)	85.08 (±0.0088)	88.64 (±0.0077)
	AE	94.72 (±0.0127)	84.18 (±0.0078)	80.09 (±0.0199)	79.67 (±0.0296)
Semi-supervised	Manifold Mixup	94.92 (±0.0012)	84.68 (±0.0279)	80.24 (±0.0652)	78.79 (±0.0135)
	VIME	95.71 (±0.0013)	84.54 (±0.0408)	81.36 (±0.0301)	79.02 (±0.1329)
	Supervised	93.69 (±0.0234)	83.19 (±0.0663)	79.63 (±0.0519)	75.95 (±0.0202)
	Self-SL	95.82 (±0.0131)	85.16 (±0.0249)	81.38 (±0.0373)	79.46 (±0.0463)
Ours (Ablation)	Self-SL+PL	97.01 (±0.0066)	85.26 (±0.0207)	81.65 (±0.0370)	79.92 (±0.0682)
	Ours	**97.58** (±0.0078)	**85.42** (±0.0210)	**81.88** (±0.0123)	**80.41** (±0.0205)

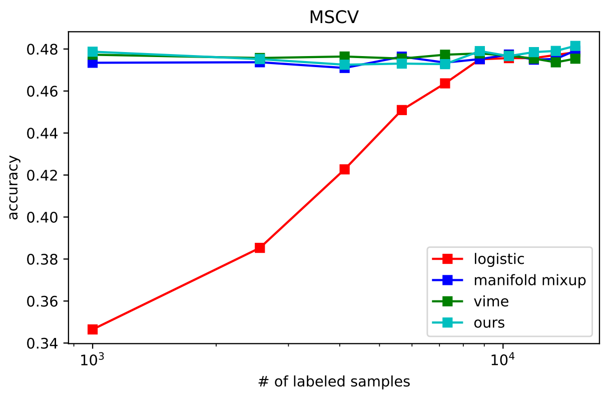
In Table 4.1 we show that our method consistently out performs previous methods. Further, through our ablation we demonstrate the various components of our framework each provide benefit in improving downstream task performance, allbeit the level of the effectiveness is dataset dependant. As a reference Catboost trained on 100% of the labeled samples is provided as well. Our results on 10% labeled MNIST with the help of pseudo-labeling we outperform this reference point.

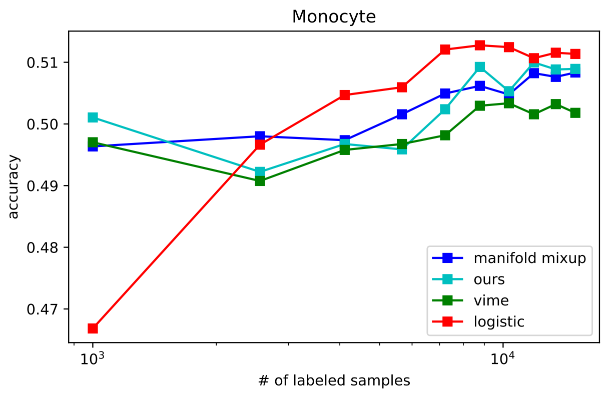
4.4.2 Genomics Datasets

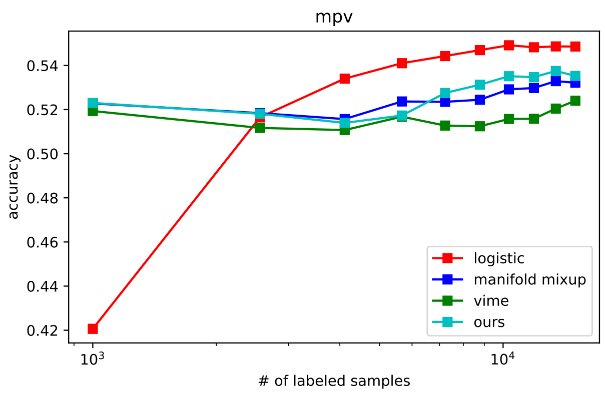
We assessed the accuracy of our method on the UK Biobank genotypes consisting of around 300,000 individuals genotyped at around 10 millions SNPs. In this experiment, we restricted our analysis to SNPs with minor allele frequency larger than 1%. Moreover, SNPs that fail the Hardy-Weinberg test at significance threshold 10^{-7} were removed. We restricted our analysis to self-reported British white ancestry individuals. We selected four complex traits measured in UK Biobank. Including all SNPs in our analysis is computationally expensive. Therefore, for every trait, we selected around 1000 SNPs with smallest p-value based on a publicly available summary statistics. Note that the set of selected SNPs is different across traits after p-value filtering.

To explore the efficacy of our semi-supervised framework on limited labeled data sets in practical setting, we compared the accuracy of our method with state of the art methods by varying the number of labeled individuals and using the remaining individuals as unlabeled samples. The results on four phenotypes are shown in Figure **??**. From these figures we can see the semi-supervised methods outperform logistic regression model for cases where we only have access to a few thousand labeled samples. For two out of the four phenotypes logistic regression model outperforms the deep supervised models when adequate labeled samples are available i.e. $> 10^4$. This may be due to only a subset of features being selected using p-value threshold and hence making deeper models prone to overfititng.


Smoking Status
accuracy
# of labeled samples
logistic
manifold mixup
vime
ours


MSCV
accuracy
# of labeled samples
logistic
manifold mixup
vime
ours


Monocyte
accuracy
# of labeled samples
manifold mixup
ours
vime
logistic


mpv
accuracy
# of labeled samples
logistic
manifold mixup
vime
ours

4.5 Conclusion

Tabular data presents a different challenge compared to images and text, as similar structure or semantics aren't present, hence mitigating the transfer-ability of methods from those domain to the tabular domain. As a result extending semi-supervised methods that work well in those domains is more challenging for the tabular domain. Additionally, as most of the literature revolves around images and text not many pre-text tasks, and transformations have been investigated for such unstructured datasets where "correlations" or semantic meanings aren't immediately present in the data. Instead, we propose a framework for extending the recent contrastive learning paradigm to the tabular domain and help propel it's success in this domain as well. We do this by mapping samples to the latent space and creating new examples interpolating between samples in this space. We empirically show the effectiveness of the proposed method, and demonstrate how it improves learning from tabular data with limited labels. Further, improvements on pre-text tasks or augmentation methods for tabular datasets will dramatically improve the applicability of deep learning for these data modalities.

CHAPTER 5

Multi-modal Over Sampling Framework for Tabular Data

Real-world binary classification tasks are in many cases imbalanced, where the minority class is much smaller than the majority class. This skewness is challenging for machine learning algorithms as they tend to focus on the majority and greatly misclassify the minority. Adding synthetic minority samples to the dataset before training the model is a popular technique to address this difficulty and is commonly achieved by interpolating minority samples. Tabular datasets are often multi-modal and contain discrete (categorical) features in addition to continuous ones which makes interpolation of samples non-trivial. To address this, we propose a latent space interpolation framework which (1) maps the multi-modal samples to a dense continuous latent space using an autoencoder; (2) applies oversampling by interpolation in the latent space; and (3) maps the synthetic samples back to the original feature space. We defined metrics to directly evaluate the quality of the minority data generated and showed that our framework generates better synthetic data than the existing methods. Furthermore, the superior synthetic data yields better prediction quality in downstream binary classification tasks, as was demonstrated in extensive experiments with 27 publicly available real-world datasets.

5.1 Introduction

Imbalanced classification tasks arise naturally, for example, consider the problem of credit card fraud detection where the vast majority of transactions are legitimate and only a few are fraudulent. This imbalance is challenging for machine learning (ML) algorithms as they tend to classify the majority well while poorly classifying the minority. This phenomenon occurs because the ML algorithms optimize a different metric than the user is interested in, resulting in an undesirable bias in the final trained model. Oversampling the minority class and under-sampling the majority class before training the ML algorithm are popular methods to address this challenge. They are effective because they yield an augmented dataset for which the algorithm's loss function and the desired loss function are more similar. See the formal description in Section 5.3.1.

As opposed to random oversampling or increasing the weight of the minority class, *SMOTE* was the first method to propose balancing the dataset by adding synthetic minority samples [85]. In *SMOTE*, the synthetic minority samples are created by interpolating pairs of the original minority points, hence instead of working in the original sample space by replicating samples, it generates new samples in the feature space. However, while effective for densely sampled feature spaces, When the feature space is sparse, the linear interpolation of samples might yield unrealistic low probability samples. Thus, *SMOTE* only interpolates pairs of points that are relatively close in the feature space. However, this strategy is inefficient when the feature space is high-dimensional, see [86].

Many real-life tabular datasets are multi-modal and include not only continuous numeric features but also categorical features e.g. gender, color, marital status, etc. We will denote the former continuous and the latter discrete. It is common to assume that the classification of each feature (continuous or discrete) is known. When interpolating samples of a multi-modal dataset with discrete features we face two challenges: (i) how to calculate distances between samples? and (ii) how to set the discrete feature values for the synthetic samples? In other words, how can we interpolate "dog" and "cat"? The original *SMOTE* paper introduced *SMOTE-NC* which is a *SMOTE* variant that supports discrete features by: (i) using a heuristic to estimate the distance implied by the discrete features and (ii) the discrete feature values are set to be the majority of the k nearest neighbors.

Since its' inception, more than 100 extensions and variants of *SMOTE* were proposed. However, to the best of our knowledge, *SMOTE-NC* is the only variant that supports discrete features. A common method used to apply any interpolation method to discrete features is to encode them using ordinal integers and consider them to be continuous, see e.g. [87]. This method results in synthetic samples that are continuous rather than discrete, thus not realistic. Moreover, many algorithms are optimized for handling discrete features (e.g. Catboost[88], lightGBM[89] and deep networks[90]) and such augmentation will render these optimizations useless.

Considering high-dimensional multi-modal data, it is commonly assumed that the data resides on an unknown lower dimensional manifold. For such sparse high-dimensional data, simple linear interpolation of samples can result in low probability synthetic samples. Motivated by this, we propose a latent space interpolation framework based on autoencoders that we name *Tabular AutoEncoder Interpolator* (*TAEI*). In summary, our method consists of a dimensionality reduction step where samples are mapped to a dense continuous latent space. Subsequently, samples of interest are interpolated in the learned latent space (using *SMOTE* or any other interpolation technique) before being mapped back to the original feature space. Since our framework interpolates points in the dense latent space, it creates more genuine synthetic samples thus improving prediction performance. Additionally, our approach shifts the challenge of the discrete features from the interpolation method to the encoder-decoder, where we can leverage previous research. Thus, by mapping the discrete and continuous features to a unified continuous latent space, we enable dozens of previously proposed smote variants and extensions to produce multi-modal data. An overview of our method is shown in Figure 5.1.

When adding minority synthetic samples to the training data in order to improve prediction quality, intuitively, it's desired that the distribution of synthetic minority samples will mimic the underlying minority samples distribution. That is: (i) synthetic points will be generated wherever the underlying minority distribution is high and (ii) will not be generated where the underlying minority distribution is low. There is a trade-off between the two requirements above: For example, an oversampler can "play it safe" and create minority samples only where it has high confidence thus satisfying (ii) but not (i). Or, an oversampler can "take a risk" and create minority samples in larger regions thus satisfying (i) but not (ii). Two metrics are defined in Section 5.4 to capture these notions: *cover* and *error* for (i) and (ii) respectively. Then, by extensive experimentation with artificial data we map the methods proposed in the literature and our framework to the *cover-error* plane. As the

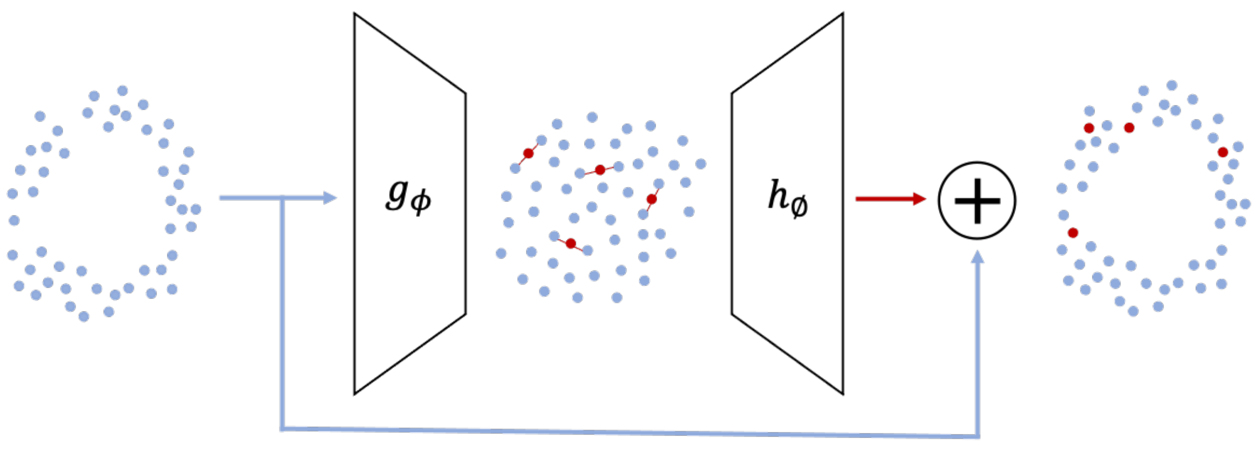


Figure 5.1: Overview of our proposed method. The sparse data is encoded into a dense latent space where synthetic samples (in red) are generated by interpolation. The interpolated samples are then decoded and added to the original data.

best *cover-error* balance is unknown, we identify the *cover-error* Pareto-optimal curve and show that the oversamplers of our framework constitutes its mid-range so other methods yield either much higher *error*, worse *cover* or sub-par performance for both. Furthermore, by experimentation with 27 real-world binary classification datasets we show that the superior quality data generated by our proposed method yields better prediction quality for the downstream tasks.

To summarize, the contributions of our work are[1]:

1. We propose a new minority oversampling framework for tabular multi-modal data.
2. We introduce novel metrics to directly measure the quality of the generated data and demonstrate the effectiveness of our approach in generating high-quality synthetic data.
3. Our framework is decomposable with any auto-encoding technique for multi-modal data and any continuous interpolation method, allowing it to take advantage of future advancements in both fields (auto-encoding and interpolation)

[1]Implementations of the oversamplers proposed here can be found at https://github.com/aws/sagemaker-scikit-learn-extension/tree/master/src/sagemaker_sklearn_extension/contrib/taei

5.2 Related Work

Since *SMOTE*'s inception, more than 100 extensions and variations have been published. Due to their large number, we will not survey all *SMOTE* variants but only the methods that resemble our approach. Moreover, to the best of our knowledge only *SMOTE-NC* proposed in the original paper supports discrete features. In fact, the two recent survey papers do not even mention discrete features or multi-modal data, see [91, 87]. [87] empirically compared 85 variants of *SMOTE* on 104 imbalanced datasets including both continuous and discrete features. The paper does not describe how the discrete features were processed. We believe that they were simply treated as continuous after being encoded as ordinal integers. The best method was found to be *Poly*[92] which, similarly to *SMOTE*, interpolates points in the feature space, but using a slightly different scheme. Differently from *SMOTE*, *Poly* allows interpolating of minority points that are not very close to each other. The second best performing algorithm, *ProWSyn*[93], also allowed interpolating of far apart minority samples.

Several variants of *SMOTE* share the idea of mapping the samples to another space which has some desired properties, interpolate in the new space and map the synthetic samples back to the original feature space. [94] proposed to map the samples using local linear embedding aiming to create a lower-dimensional space where the data is more separable. [95] similarly proposed to use isometric feature mapping (Isomap). Kernel functions were also used to map the features[96, 97]. When the classifier is an SVM, oversampling could be done directly in the kernel space[98].

As they are very natural, autoencoders were previously proposed as a means to provide the bi-directional mapping. [99] proposed to create synthetic samples by adding Gaussian noise to the original samples in the latent space and then decode them back to the feature space. Later, the same authors proposed to apply *SMOTE* in the latent space, see [100]. However, there are two key differences between their approach and ours: (1) they train the autoencoder on the minority class samples only which, due to the low number of samples, force them to train shallow models and (2) they did not consider discrete features. [101]

proposed to encode the samples and train the classifier in the latent space. To improve unsupervised anomaly detection, [102] proposed to use adversarial auto encoders to encode the features into a Gaussian mixture latent space and interpolate samples near the boundaries of the distribution. Some methods incorporate mapping the feature to other spaces but use the new space differently. MOT2LD[103] first maps each training sample into a low-dimensional space and then applies clustering and weighting heuristics in the low-dimensional space. In ADOMS[104] each sample neighbors are derived in the original feature space, however, the synthetic sample is then created along with the first principal component of the k neighbors.

Synthetic data generation has been studied extensively in the context of deep learning for image collections. For example, [105] is an interpolating framework very similar to SMOTE intended for deep networks. Interpolation in the latent space using autoencoders was also proposed [106, 107]. Similarly to our work, the quality of the resulting synthetic samples was directly studied in [108] using artificial data. However, the study on image collection is not easily transferable to tabular datasets as image collections have different characteristics. For example, image data is spatially coherent. Furthermore, image collections usually contain hundreds of thousands of images while the number of samples in our tabular datasets is usually two or three orders of magnitude smaller. In fact, the median and average number of samples in our dataset collection are about $10k$ and $3k$ respectively.

Nevertheless, inspired by study of image collections, generation of tabular multi-modal data was studied in the context of generative adversarial networks (GANs). The challenge of synthesizing the discrete features was addressed using three methods: noising the discrete data [109], using a Gumbel softmax [110, 111] or using an autoencoder to map the data to a continuous latent space and train the GAN in that space [112]. FAST-DAD [113] generates multi-modal synthetic samples by augmenting existing samples using Gibbs sampling. The Gibbs based augmentation method requires a pre-trained conditional expectation model for all features and another model is used to label the new samples.

5.3 Method

5.3.1 Oversampling Overview

Consider a dataset $(X, Y) = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i are observations sampled from a data-generating distribution $p(x)$ and $y_i \in \{0, 1\}$. We consider a multi-modal setting where x_i is a concatenation of discrete $\mathcal{D} = [D_1, \cdots, D_{|\mathcal{D}|}]$ and continuous features $\mathcal{C} = [C_1, \cdots, C_{|\mathcal{C}|}]$. In classification tasks, the goal is to obtain a classifier function $f : \mathcal{X} \to \mathcal{Y} \in \mathcal{F}$ optimizing a given loss function $l(\cdot, \cdot)$

$$f = \min_{f \in \mathcal{F}} l\left(f(X), Y\right)$$

Where $f(X)$ is the vector of predictions or class probabilities and Y is the vector of true labels. For imbalanced binary classification, $l(\cdot, \cdot)$ is commonly ROC-AUC, F_1-score or G-score which are not additive (as defined below).

We consider the common case of obtaining the classifier f by training an ML algorithm. To the best of our knowledge, all mainstream ML algorithms assume additivity of the loss. Formally, they optimize an additive loss function l_A such that:

$$l_A\left(f(X), Y\right) = \sum_{i=1}^{N} l'_A(f(x_i), y_i)$$

Where l_A is commonly logistic loss, cross entropy or hinge loss. The challenge stems from the desired loss $l(\cdot, \cdot)$ being different from the loss the algorithm optimizes $l_A(\cdot, \cdot)$. Oversampling schemes address this challenge by adding minority class synthetic samples (X', Y') to the dataset aiming at making f_A^* that minimizes $l_A\left(f(X \cup X'), Y \cup Y'\right)$ more similar f^* that minimizes $l\left(f(X), Y\right)$.

5.3.2 Method Overview

Our proposed framework is based on the standard autoencoder scheme proposed in [41]. As we are concerned with multi-modal data, for every categorical column D, we introduce an embedding layer $\mathcal{W} \in \Re^{|D| \times d_{|D|}}$. The input which is a concatenation of discrete and continuous features is passed through the embedding layer resulting in a feature vector

$x_i \in \Re^{|\mathcal{C}|+\Sigma_i^{|D|} d_{|D_i|}}$ that is used as the input to the autoencoder

$$z = g_\phi(x)$$

Where $g_\phi(\cdot)$ is a high capacity deep neural network, such as a set of fully connected layers.

As in traditional autoencoders, a decoder module $h_\theta(z)$ is used to map the latent points back to the feature space x. The objective minimized while training the autoencoder is the reconstruction loss:

$$\min_{\theta,\phi} E_{p(x)}[d(h_\theta(z), x)]$$

In a multi-modal setting, the reconstruction contains both discrete and continuous variables therefore the function $d(\cdot)$ is a sum of the softmax loss and mean squared error (MSE):

$$J_{recon}(D; \theta, \phi) = \sum_{x_i} \sum_{c}^{|C|} ||h_\theta(z_i)^c - x_i^c||_2^2 + \alpha \sum_{x_i} \sum_{d}^{|D|} \sum_{o} \mathbf{1}[x_i^d = o] \log(h_\theta(z_i)^o)$$

Where α is a constant controlling the weight of the softmax loss. In our experiments $\alpha = 0.3$ was used.

Once we have a fully trained autoencoder we leverage existing interpolation methods to synthesize minority samples in the latent space and using the decoder to map them back to the feature space. In our experiments *SMOTE* and *Poly*[92] were used. *Poly* is a simple interpolation technique that was selected based on it's superior performance in the experiments of [87]. Nevertheless, our framework and the published code support any underlying interpolation method.

5.3.3 Autoencoder Schemes

As samples are interpolated in the latent space, it is desired that the latent space manifold will be dense and without holes i.e. without low probability regions surrounded by high probability ones. When the latent space manifold contains holes, pairs of points from high probability regions might by interpolated into these low probability regions which can result in the generation of unrealistic samples. Vanilla autoencoders do not have such guarantees, hence we studied several other autoencoders with the motivation of creating a dense latent space in which linear interpolation of samples from the manifold will reside on the manifold as well. Formal description of the models described below can be found in Appendix **??**.

Auto Encoder (AE) We compose both our encoder g_ϕ and decoder h_θ as fully connected layers FC-BN-ReLU with depth 3 and the size of all layers was either 100 or 200. The latent representation dimension was 20 or 40. For each discrete feature, the embedding size is set to $min(600, round(1.6 * |D|^{0.56})$ where $|D|$ is the number of unique values for the discrete column.

The latent space representation for a sample x is given by $z = g_\phi(x)$. It is decoded back by $x' = h_\theta(z)$. The objective minimized while training the autoencoder is the reconstruction loss:

$$E_{p(x)}[J_{recon}(h_\theta(g_\phi(x)), x)]$$

Where

$$J_{recon}(X;\theta,\phi) = \sum_{x_i}\sum_{c}^{|C|} ||h_\theta(z_i)^c - x_i^c||_2^2 + \alpha \sum_{x_i}\sum_{d}^{|D|}\sum_{o} \mathbf{1}[x_i^d = o]\log(h_\theta(z_i)^o)$$

Where α is a constant controlling the weight of the softmax loss. In our experiments $\alpha = 0.3$ was used. AE oversampler is provided by *AEOversampler* of our package.

Variational autoencoder (VAE) Due to their probabilistic nature, VAEs naturally yield a smooth latent space manifold. We have used the vanilla VAE[114] with embeddings for the discrete columns and J_{recon} as defined above.

We have used the vanilla VAE[114] with embeddings for the discrete columns as described above. The encoder and decoder an reconstruction loss are the same as for AE. VAE oversampler is provided by *VAEOversampler* of our package.

Regularized autoencoder (RAE) Recent developments have shown that adding a regularization loss in the latent space results in a deterministic autoencoder with similar properties as in VAEs [115], all be it much faster and easier to train. Thus, in RAE, norm 2 regularization is added to the latent space representation (z) and to the decoder weights.

RAE is based on our AE with regularization on the latent space i.e., instead of optimizing only J_{recon} we optimize

$$E_{p(x)}[J_{recon}(h_\theta(g_\phi(x)), x) + r_1 \cdot ||z||_2^2 + r_2 \cdot \sum_{w \in h_\theta} ||w||]$$

where w are the weights of the decoder layers and $|| \cdot ||$ is the Frobenius norm. r_1 and r_2 are the weights for these regularization terms. In our experiments we used $1e-3$ for both. RAE oversampler is provided by *RAEOversampler* of our package.

Adversarial autoencoder (AAE) Similarly to [102], we train the autoencoder in an adversarial setting vs a discriminator in the latent space. The discriminator tries to classify real samples latent representations vs vectors drawn randomly from a Gaussian distribution and the autoencoder aims at minimizing J_{recon} while fooling the discriminator.

AAE is based on our AE with regularization on the latent space using an adversarial scheme. A discriminator was added, a classification net with the same structure as the decoder with the task of predicting whether a latent space representation is of a real point or sampled from a Gaussian distribution. Backward propagation for the encoder-decoder network using the reconstruction loss and for the discriminator loss were run intermittently with the same learning rate. AAE oversampler is provided by *AAEOversampler* of our package.

Interpolate adversarial autoencoder (IAAE) IAAE is introduced in order to directly penalize invalid interpolation in the latent space. It is achieved in an adversarial setting using a discriminator with the task of classifying real samples latent space representations vs latent space interpolated point. In order to fool the discriminator, the encoder learns a latent representation in which interpolated samples are similar to real ones

IAAE is based on our AE with regularization on the latent space using an adversarial scheme. A discriminator was added, a classification net with the same structure as the decoder with the task of predicting whether a latent space representation is of a real point or results from an interpolation of two samples in the latent space. Backward propagation for the encoder-decoder network using the reconstruction loss and for the discriminator loss were run intermittently with the same learning rate. At each of the discriminator's back propagation steps, half of the samples were randomly interpolated with the other half in the latent space to create the discriminator "false" samples. The discriminator task was to separate these interpolated "false" samples from the real ones. IAAE oversampler is provided by *IAAEOversampler* of our package.

Adversarially constrained autoencoder interpolation (ACAI) Adaptation of the autoencoder proposed in [108] for tabular datasets. The autoencoder is regularized by mixing two samples with ratio α and $1-\alpha$ in the latent space. The adversarial critic network tries to predict α given only the decoded interpolated sample.

Adaptation of the autoencoder proposed in [108] for tabular datasets. The autoencoder is regularized by mixing two samples with ratio α and $1-\alpha$ in the latent space. The adversarial critic network tries to predict α given only the decoded interpolated sample. ACAI is based on our AE with a critic regression net with the same structure as the decoder. At each of the critic's back propagation steps, half of the samples of the batch were randomly mixed with the other half using a random vector α with components i.i.d selected uniformly from $[0, 0.5]$. The interpolated samples were decoded back to the feature space where the critic predicted the αs. Backward propagation for the encoder-decoder network using the reconstruction loss and for the critic loss were run intermittently with the same learning rate. ACAI oversampler is provided by *ACAIOversampler* of our package.

Training the autoencoders When training these networks all samples in the training set are used (minority & and majority samples). In all experiments Adam optimizer[34] was used to train the autoencoders with an initial learning rate of $1e-3$ decayed every epoch by a factor of $5e-3$ and trained for a maximum of $10k$ epochs (recall that our datasets are small) with early stopping on the loss of the validation set. For the adversarial autoencoder, the same learning rate and decay were used to train the discriminator network.

5.4 Evaluation on Artificial Data

In the spirit of [108], we use artificial datasets to evaluate the quality of the synthetic minority samples generated by the oversamplers. As opposed to real datasets, for artificial ones we know the underlying distribution thus we can directly measure the quality of the synthetic data. Generally, artificial datasets are created by defining the minority and majority distributions in feature space and sampling from them. In our experiments these distributions are defined by two non-overlapping manifolds: the *minority manifold* and the *majority manifold*. The artificial datasets are created by sampling uniformly from these two manifolds. To create multi-modal data, the artificial samples are partially discretized as described below.

To create artificial sparse data, we (1) define a low-dimension manifold in feature space and (2) split the manifold into *minority manifold* and *majority manifold*. So it is guaranteed that the minority and majority samples reside in separate regions of the same low-dimension manifold. For a d-dimensional feature space, the manifold is a d-dimensional unit sphere (which is a $(d-1)$-dimensional manifold). The *minority manifold* is a thin slice of the sphere. Formally, for a D-dimensional feature space the *minority manifold* is defined by $||X|| = 1 \wedge |x_0| \leq \alpha$ and the *majority manifold* is given by $||X|| = 1 \wedge |x_0| > \alpha$ where $||\cdot||$ is the euclidean norm and x_0 is the first coordinate of the vector X. α is a constant set so the density of both manifolds will be the same. For example, when the minority samples are 5% of the samples as was used in the experiments, $\alpha = 0.06$.

		Cov	***Err***
Pareto-optimal	TGAN	0.323	0.617
	Poly+VAE	0.335	0.301
	SMOTE+VAE	0.355	0.138
	SMOTE+AE	0.378	0.082
	No OS	0.449	0.000
	CTGAN	0.355	0.683
	Poly+RAE	0.364	0.211
	Poly+ACAI	0.367	0.225
	Poly+AE	0.370	0.187
	SMOTE+RAE	0.378	0.094
	SMOTE/NC	0.380	0.121
	SMOTE+ACAI	0.384	0.110
	Poly+IAAE	0.388	0.321
	Poly+AAE	0.395	0.393
	SMOTE+AAE	0.400	0.315
	SMOTE+IAAE	0.414	0.264

Table 5.1: *

(a) All

		Cov	***Err***
Pareto-optimal	**Poly+VAE**	0.337	0.187
	SMOTE+VAE	0.364	0.102
	Poly	0.370	0.063
	SMOTE+AE	0.393	0.044
	SMOTE/NC	0.405	0.025
	No OS	0.463	0.000
	TGAN	0.342	0.622
	CTGAN	0.367	0.673
	Poly+RAE	0.377	0.140
	Poly+ACAI	0.381	0.166
	Poly+AE	0.389	0.097
	SMOTE+RAE	0.398	0.054
	Poly+IAAE	0.407	0.292
	SMOTE+ACAI	0.408	0.075
	SMOTE+IAAE	0.421	0.221
	Poly+AAE	0.430	0.362
	SMOTE+AAE	0.433	0.302

Table 5.2: *

(b) Continuous

		Cov	***Err***
Pareto-optimal	TGAN	0.303	0.612
	Poly+VAE	0.333	0.414
	SMOTE+VAE	0.347	0.174
	SMOTE+RAE	0.357	0.134
	SMOTE+AE	0.362	0.120
	No OS	0.435	0.000
	CTGAN	0.342	0.694
	Poly+RAE	0.351	0.283
	Poly+AE	0.352	0.277
	Poly+ACAI	0.353	0.285
	SMOTE/NC	0.355	0.217
	SMOTE+ACAI	0.360	0.144
	Poly+AAE	0.360	0.424
	SMOTE+AAE	0.367	0.328
	Poly+IAAE	0.369	0.351
	SMOTE+IAAE	0.406	0.307

Table 5.3: *

(c) Multi-modal

In Table 5.4. *Cover* and *error* of the oversamplers averaged over artificial datasets. (a) results averaged over all artificial datasets; (b) results averaged over all continuous artificial datasets; (c) results averaged over all multi-modal artificial datasets. The oversamplers of the Pareto-optimal front are above the mid-line and our proposed methods are in bold.

To create a continuous artificial dataset of size N with μ of the samples being minority: (1) sample uniformly $N(1-\mu)$ majority samples from the *majority manifold* and $N\mu$ minority samples from the *minority manifold*, and (2) rotate the sphere randomly. To create a multi-modal dataset with both continuous and discrete features: (1) create a continuous artificial dataset, (2) discretize some of the features using M bins equally spaced between -1 and 1, and (3) randomly change the order of bins. The order of bins is randomized in order to remove the geometric information from the ordinal representation i.e., avoid bin 0 being the leftmost, 1 being the one next to it and so on. We experimented with $D \in [6, 10]$, $N \in [1k, 10k, 100k]$, $M = 7$ and $\mu = 0.05$ so we had 6 continuous and 6 multi-modal classes of artificial datasets. In the multi-modal datasets half of the features were discretized. In our experiments, 7 artificial datasets were generated for each of the 12 classes using different random seeds and the results were averaged over them.

As previously discussed, when synthesising minority data for classification, it's desired that the distribution of synthetic minority samples will mimic the underlying minority samples distribution. That is: (i) synthetic points will be generated wherever the underlying minority distribution is high and (ii) will not be generated where the underlying minority distribution is low. However, generating synthetic minority samples in regions in feature space that have low minority density and high majority density is more harmful than generating them in regions with low minority and majority densities. Thus, we reformulate (ii) as: do not generate minority samples in regions where the majority density is higher than the minority density. The metrics used to measure these notions are formally defined as:

***Cover* (*Cov*)** measures how well the synthetic minority samples cover the *minority manifold*. *Cover* is calculated by (1) using the oversampler to create $10k$ synthetic minority samples, (2) sampling $500k$ minority samples from the *minority manifold*, (3) for each "real" minority sample, calculate the distance to the closest synthetic point, (4) cover is defined as the average of these distances. Note that for *cover*, lower is better.

***Error* (*Err*)** quantifies how much the synthetic minority samples are reliable as minority. It is calculated by (1) using the oversampler to create $10k$ synthetic minority samples, (2) sampling $500k$ minority samples from the *minority manifold* and $500k$ majority samples from the *majority manifold*, (3) for each synthetic sample find the closest "real" sample. A synthetic sample is *invalid* if the closest "real" point is a majority point. *Error* is defined as the ratio of invalid synthetic points to the number of synthetic points generated.

Both *cover* and *error* rely on a distance metric in the feature space. Euclidean distance was used for continuous feature spaces. For multi-modal, as they have an underlying geometry, the discrete features where converted back to the continuous space before calculating the Euclidean distance. It was achieved by replacing each discrete feature value with the center of the bin previously used for discretizing. It was done for both the generated synthetic samples and the "real" ones.

As previously discussed, there is a trade-off between optimizing *cover* and *error*. An oversampler can "play it safe" and create minority samples only where it has high confidence thus yielding a lower *error* but also a higher (worse) *cover*. On the other hand, an oversampler can "take a risk" and create minority samples in larger regions thus lowering (improving) *cover* but increasing the *error*. Since the preffered balance is unclear, and the optimal *cover-error* ratio probably varies for different datasets, we are interested in the Pareto-optimal front.

Recall that there are very few methods that allow generation of multi-modal tabular data including both continuous and discrete features. We compared our autoencoder based oversamplers to all available methods:

SMOTE/NC: As *SMOTE* supports only continuous datasets and *SMOTE-NC* supports only datasets that have both numeric and discrete features (fails otherwise) we paired them together under the name *SMOTE/NC*.

CTGAN[111]: A recent generative GAN model specifically designed to handle tabular datasets by conditioning on discrete columns. The open source implementation was used[2].

TGAN[116]: The TGAN network trained using Wassertein GAN loss. The open source implementation was used[3].

Poly: An interpolation method that supports only continuous features thus was included only in the experiments with continuous artificial datasets. *Poly* was included as it performed the best amongst the 85 *SMOTE* variations evaluated in [87].

No OS: Not oversampling - using the original minority points. In addition to providing a baseline, the random oversampling used in Section 5.5 yields the same *cover* and *error* as *No OS*.

As CTGAN and TGAN generate both majority and minority samples, they were used by: (1) considering the label to be an additional discrete feature in the training phase; (2) when oversampling, generate more samples then required and filter out the majority samples; (3) repeat the previous step until the desired number of minority samples is generated.

The resulting *Cover* and *error* of the oversamplers are presented in Table 5.4. The Pareto-optimal oversamplers are above the mid-line. The results averaged over all artificial datasets can be found in Table 5.4a and Figure 5.2. The results averaged over several slices of the artificial datasets can be found in Tables 5.4b, 5.4c and 5.4.

Overall, TGAN yields the best *cover*. However, this comes with the price of very high *error*. *No OS* yields the worst *cover* but with an *error* of zero, hence it's always a part of the Pareto-optimal front. From our proposed autoencoders, the VAE and AE based oversamplers consistently outperform the others. Overall (Table 5.4a) VAE and AE fill the mid-range between TGAN and *No OS* and outperform all other oversamplers in that range. *Poly* supports only continuous features hence it was omitted for multi-modal datasets. However, for continuous datasets it performs well and belongs to the Pareto-optimal front, see Table 5.4b.

[2]https://github.com/sdv-dev/CTGAN

[3]https://github.com/Baukebrenninkmeijer/On-the-Generation-and-Evaluation-of-Synthetic-Tabular-Data-using-GANs

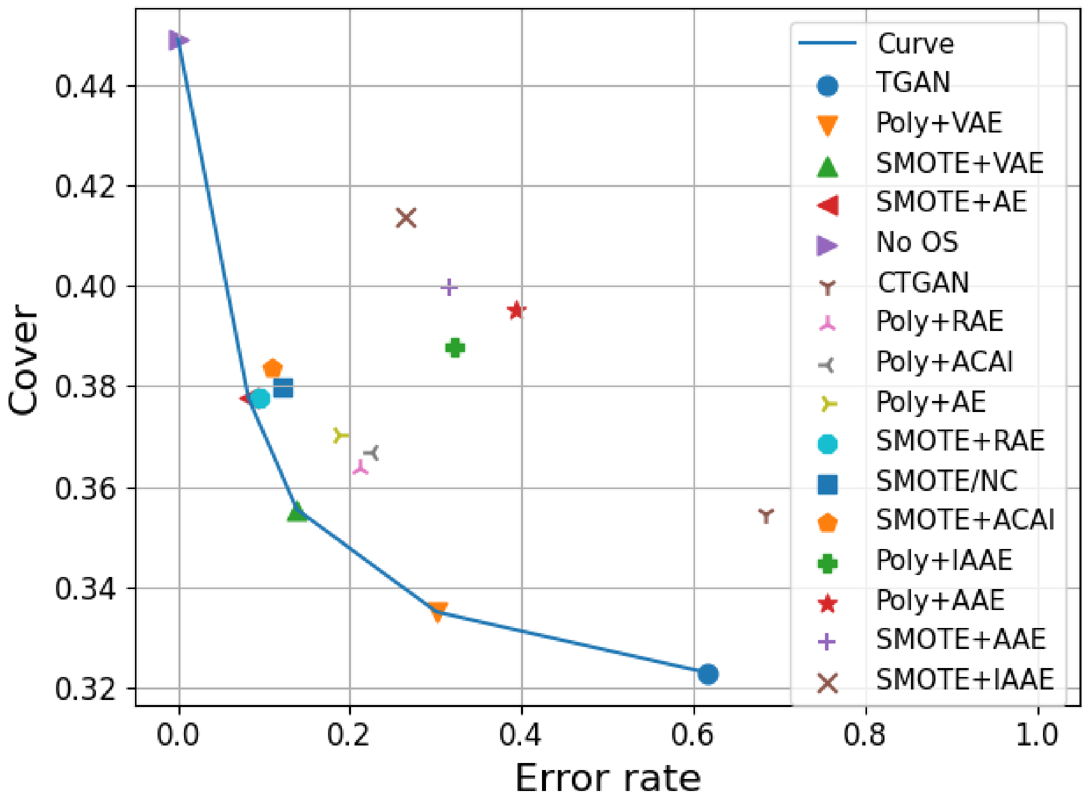


Figure 5.2: *Cover* and *error* for the oversamplers averaged over all artificial datasets. The Pareto-optimal curve is presented by a blue line.

We postulated that the GAN based methods, TGAN and CTGAN, would benefit from training on a large datasets. However, this was not the case as the results are consistent for large and small datasets, see Table 5.4 for the results on artificial datasets comprising $100k$ samples.

An example of our framework generating more realistic synthetic data compared to *SMOTE* is presented in Figure 5.5.

5.5 Evaluation on Real-World Data

		Cov	***Err***
Pareto-optimal	**SMOTE+VAE**	0.265	0.084
	SMOTE+RAE	0.274	0.043
	SMOTE+AE	0.275	0.031
	No OS	0.287	0.000
	SMOTE/NC	0.276	0.061
	Poly+VAE	0.283	0.326
	SMOTE+ACAI	0.290	0.053
	Poly+RAE	0.302	0.204
	Poly+ACAI	0.306	0.162
	SMOTE+AAE	0.311	0.260
	TGAN	0.314	0.617
	Poly+AE	0.323	0.153
	CTGAN	0.339	0.645
	Poly+AAE	0.360	0.357
	SMOTE+IAAE	0.385	0.361
	Poly+IAAE	0.391	0.386

Table 5.4: *Cover* and *error* of the oversamplers averaged over artificial datasets with $100k$ samples. Our proposed methods are in bold.

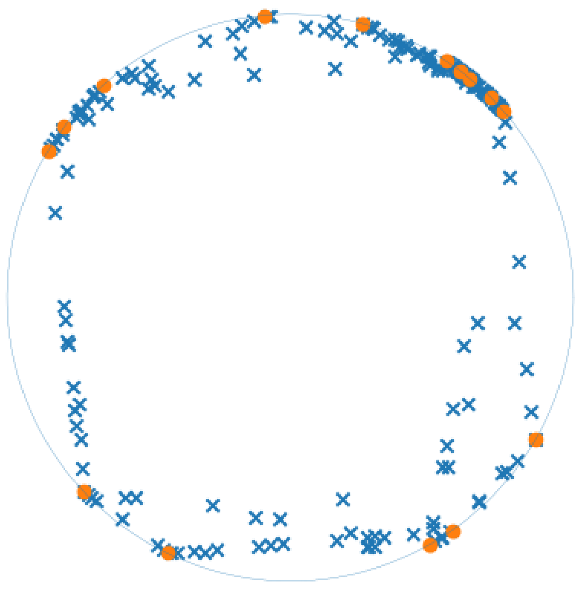


Figure 5.3: *

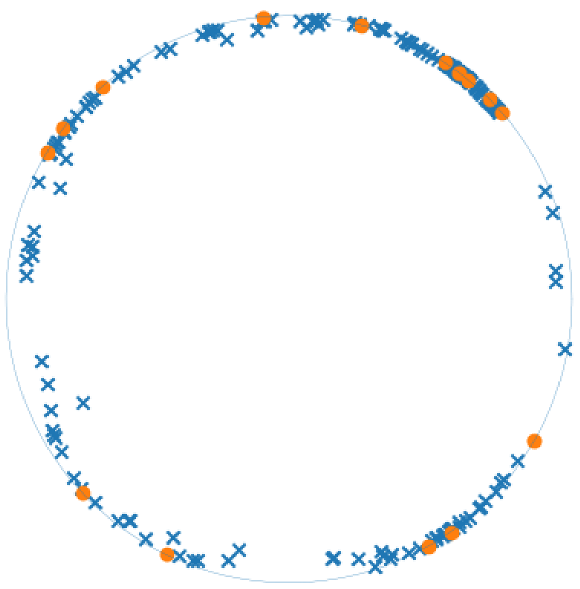


Figure 5.4: *

(a) *SMOTE* (b) Our framework

Figure 5.5: An example of the synthetic minority samples generated by *SMOTE* and our framework. The artificial manifold is a $3d$ sphere thus the minority manifold resembles a $2d$ ring. The real minority samples are marked by orange dots and the generated synthetic samples by blue $\times$s. See (a) where *SMOTE* generates some low probability sample inside the ring. On the other hand, see (b), where our framework by interpolating samples in the dense latent space generates realistic samples i.e., closer to the underlying minority manifold, on the ring as opposed to in it.

	abalone	**abalone_19**	**arrhythmia**	**car_eval_34**	**car_eval_4**	**coil_2000**	**ecoli**	**isolet**
No OS	0.8617	0.733	0.968	0.9973	0.9985	0.745	0.948143	0.9913
CTGAN	0.8617	0.713	0.981	0.9978	0.9987	0.749	**0.96554**	0.9916
Poly	0.8632	0.767	**0.987**	0.9985	**0.9996**	0.746	0.960856	0.9918
Poly+AE	0.8621	0.769	0.975	0.9978	0.9989	0.753	0.959518	**0.9924**
Poly+VAE	0.8617	**0.79**	0.985	**0.9988**	0.9995	**0.756**	0.96554	0.9916
ROS	0.8632	0.738	0.983	0.9979	0.9993	0.749	0.94898	0.9919
SMOTE	0.8632	0.753	0.985	0.9984	0.9989	0.746	0.963031	0.9919
SMOTE+AE	**0.8635**	0.773	0.985	0.9981	0.9992	0.752	0.960355	0.9918
SMOTE+VAE	0.863	0.749	0.984	0.9981	0.9989	0.755	0.960522	0.9912
SMOTE-NC	0.863	0.735	0.983	-	-	0.748	-	0.9921
TGAN	0.8618	0.76	0.978	0.998	0.9985	0.748	0.962529	

Table 5.5: Comparison of oversampling methods on 27 imbalanced datasets. The methods of our framework and the best scores are in bold.

	letter_img	**libras_move**	**mammography**	**oil**	**optical_digits**	**ozone_level**	**pen_digits**
No OS	0.99956	0.981	0.9557	0.9	0.999	0.891	0.99978
CTGAN	0.99958	0.987	0.9568	0.912	0.999	0.895	0.99983
Poly	0.99963	0.988	0.9578	0.932	0.9992	0.909	0.99985
Poly+AE	**0.99975**	**0.994**	0.9579	0.917	0.9992	0.905	**0.99986**
Poly+VAE	0.99962	0.992	**0.9588**	0.912	0.9992	**0.915**	0.99984
ROS	0.9995	0.986	0.9576	0.929	0.9992	0.901	0.99982
SMOTE	0.99955	0.99	0.956	**0.944**	**0.9993**	0.904	0.99982
SMOTE+AE	0.99966	0.988	0.9587	0.938	0.9991	0.899	0.99983
SMOTE+VAE	0.9996	0.992	0.9583	0.919	0.9991	0.908	0.99984
SMOTE-NC	-	-	-	0.935	0.9992	-	-
TGAN	0.99957	0.985	0.9558	0.902	0.999	0.878	0.99981

Table 5.6: Comparison of oversampling methods on 27 imbalanced datasets. The methods of our framework and the best scores are in bold.

	protein_homo	**satimage**	**scene**	**sick_euthyroid**	**solar_flare_m0**	**spectrometer**
No OS	0.9922	0.955	0.766	0.975	0.819	0.959
CTGAN	0.993	0.959	0.782	0.975	0.827	0.976
Poly	**0.9935**	0.961	0.775	0.969	0.816	0.981
Poly+AE	0.9931	0.96	0.783	**0.976**	0.817	0.973
Poly+VAE	0.9922	0.962	**0.799**	0.975	**0.829**	0.971
ROS	0.9925	0.961	0.782	0.973	0.818	0.959
SMOTE	0.9925	**0.963**	0.778	0.972	0.811	0.97
SMOTE+AE	0.9934	0.962	0.786	0.975	0.814	0.973
SMOTE+VAE	0.9919	0.961	0.795	0.975	0.815	**0.982**
SMOTE-NC	-	-	-	0.975	-	-
TGAN	0.9913	0.959	0.772	0.975	0.818	0.98

Table 5.7: Comparison of oversampling methods on 27 imbalanced datasets. The methods of our framework and the best scores are in bold.

	thyroid_sick	**us_crime**	**webpage**	**wine_quality**	**yeast_me2**	**yeast_ml8**
No OS	0.9917	**0.9242**	0.969	0.847	0.897	0.593
CTGAN	0.9965	0.92415	0.969	0.848	0.904	**0.621**
Poly	0.9965	0.92023	0.978	0.847	0.904	0.6
Poly+AE	0.9956	0.92159	0.98	0.849	0.905	0.607
Poly+VAE	**0.9971**	0.92071	0.976	**0.855**	0.906	0.618
ROS	0.9965	0.92051	0.979	0.854	**0.913**	0.595
SMOTE	0.9966	0.9213	0.973	0.848	0.902	0.612
SMOTE+AE	0.9958	0.91994	**0.981**	0.854	0.904	0.609
SMOTE+VAE	0.9968	0.92049	0.977	0.854	0.901	0.6
SMOTE-NC	0.9969	-	-	-	-	-
TGAN	0.995	0.92172	0.963	0.835	0.903	0.598

Table 5.8: Comparison of oversampling methods on 27 imbalanced datasets. The methods of our framework and the best scores are in bold.

In this section we carry out experiments to compare the effectiveness of different oversamplers in addressing imbalanced binary classification challenges. To summarize, it is done by using the various oversamplers to augment the training fold before training the classifier and comparing the classification quality on the test fold.

Data We evaluated the oversampling methods on the 27 public datasets included in *imbalanced-learn*[4][117], with a varying number of samples, dimensions, and ratios of continuous/discrete features. The preprocessing comprised of normalizing the continuous features to have a unit variance and encoding the discrete features as ordinal integers. The preprocessing was done using *RobustOrdinalEncoder* and *RobustStandardScaler* from *sagemaker-scikit-learn-extension*[5].

Metric ROC-AUC (area under the receiver operating characteristic curve) was used due to it's popularity for binary classification tasks.

Oversamplers We experiment with the oversamplers listed in Section 5.4 with a few modifications: As *Poly* and *SMOTE* do not support discrete features, we applied the common method of encoding the discrete features as ordinal integers and considered them to be continuous, see e.g. [87]. In that case, these features were marked as continuous for the classifier. As *SMOTE-NC* supports only datasets that have both numeric and discrete features (fails otherwise), we did not run it for all datasets and it was not included in the summary tables. As the VAE and AE based autoencoders oversamplers performed the best in the synthetic data experiments, we experimented with only them. An additional oversampler introduced is the random oversampler (*ROS*) which augments the dataset by adding random duplicates of samples from the existing minority samples.

Method Each dataset was randomly stratified split into training, validation and test folds with ratios 60%, 20% and 20% respectively. To evaluate the oversampling methods, the training fold was oversampled using each of the oversamplers. The oversamplers that require training, were trained on the training fold with early stopping on the validation fold when possible. Catboost[88] was used as the classifier due to its popularity and strong

[4]https://imbalanced-learn.org/stable/

[5]https://github.com/aws/sagemaker-scikit-learn-extension/

performance on multi-modal tabular data. Catboost was trained on the augmented training fold with early stopping on the (not-augmented) validation fold. Finally, ROC-AUC scores for Catboost's predictions on the validation and test folds were calculated. For each triplet of {dataset, oversampler, HPs} the experiment was repeated 7 times with different random seeds and different data splits and the ROC-AUC scores were averaged over these 7 runs.

Hyper-parameters (HPs) For each oversampler, we considered several HP configurations including the number (or ratio) of synthetic samples to generate which was 10%, 20% or 30%. For example, *Poly+VAE* had 24 HP configurations, CTGAN and TGAN had 18 each and *SMOTE* had 9.

The choice of HPs can greatly effect the experiment results. A common practice is to use the set of HPs that provides the best results. This practice is explicitly mentioned in [87] and is frequently implicitly employed e.g., [99, 102]. It is reasonable, for example, when the scientist have some previous experience with similar data and knows a-priori how to properly set the HPs. ROC-AUC scores for all oversamplers on all datasets utilizing this practice are displayed in Table 5.8 and aggregation over all datasets is displayed in Table 5.11a. When no such prior knowledge exist, it is common to select for each dataset and oversampler the HP configuration that maximizes ROC-AUC on the validation fold (and use it to calculate ROC-AUC scores on the test fold). Aggregated results with HPs selected in that manner are presented in Table 5.11b.

When using the best HPs (Table 5.11a), the top four oversampler are the VAE and AE based methods. They are followed by the simple *Poly*, *SMOTE* and the random oversampler (*ROS*). *CTGAN* performs slightly worse than *ROS* but still better than training without any oversampling (*No OS*). Finally, oversampling using *TGAN* yields worse results than *No OS*. When selecting HPs based on the validation fold scores (Table 5.11b), the top performing oversamplers are *Poly+AE* and the two VAE based oversamplers. Other than that, only *ROS* provides better prediction quality than the baseline (*No OS*).

	ROC-AUC
Poly+VAE	0.923149
SMOTE+AE	0.921254
SMOTE+VAE	0.920189
Poly+AE	0.920062
Poly	0.920055
SMOTE	0.919713
ROS	0.917974
CTGAN	0.917878
No OS	0.913276
TGAN	0.912950

Table 5.9: *

(a) Best HPs

	ROC-AUC
Poly+VAE	0.916506
Poly+AE	0.915963
SMOTE+VAE	0.913652
ROS	0.913593
No OS	0.913276
SMOTE+AE	0.913224
Poly	0.912299
CTGAN	0.911949
SMOTE	0.908617
TGAN	0.902926

Table 5.10: *

(b) Best validation HPs

Table 5.11: ROC-AUC averaged over all 27 datasets. In (a) the best oversampler HPs, for each dataset, were used. In (b) the oversampler HPs that yielded the best score on the validation fold, for each dataset, were used. Methods below the mid-line performed worse than than the baseline which is training on the original training data (*No OS*). Our proposed methods are in bold.

5.6 Discussion

Comparison of Tables 5.11a and 5.11b reveals that oversampler HPs have a major effect on prediction quality. With best HPs, all but a single oversampler yield a better prediction quality than the baseline (*No OS*) while with best validation HPs, *No OS* was ranked 5$^{\text{th}}$ among 10. Moreover, in Table 5.11a the best model improves ROC-AUC by 0.009 compared to *No OS* while in Table 5.11b, the improvement of the best model was only 0.003.

The autoencoder based approaches using VAE and AE outperformed all other methods in both scenarios with the only exception being *ROS* slightly outperforming *SMOTE+AE* in the best validation HPs scenario. So the balance between *cover* and *error* these methods provide, demonstrated in the artificial datasets study, improves prediction quality for real datasets. Also note that the wrapped oversamplers outperformed their underlying method in all cases. Specifically, *SMOTE+AE* and *SMOTE+VAE* outperform *SMOTE* and *Poly+AE* and *Poly+VAE* outperform *Poly*.

Other than our autoencoder based approaches, the only method that consistently outperformed the baseline is the simple random oversampler (*ROS*). While not improving *cover*, *ROS* provides better prediction quality by increasing the relative weight of the minority samples in the training set. Surprisingly, the relatively simple *SMOTE* and *Poly* did not improve over the baseline without optimized HPs. Note that our experiments agree with the results of [87] in that *Poly* demonstrates better performance than *SMOTE* using both HP selection practices.

The advanced GAN based methods recently proposed, CTGAN and TGAN, performed poorly even when compared to the simple interpolation methods. In the artificial data experiments these methods yielded a good *cover* with very high *error* which implies that they tend to generate many synthetic minority points in feature space regions that "belong" to the majority class. Recall that these are general generative methods which are not intended for oversampling the minority and that the label was used as a discrete feature during training. Perhaps better results can be achieved by emphasising the label during model training. Moreover, due to the complexity of these methods, training them requires considerably more hardware and is more expensive compared to our simpler autoencoder based method. In particular TGAN is the slowest and most resource hungry amongst all tested methods, due to the large set of parameters contained within each head representing different columns. TGAN failed training on *isolet* dataset, probably because it includes 617 features - the most in our dataset collection.

5.7 Conclusion

Little attention has been devoted towards generation of synthetic multi-modal tabular data including both continuous and discrete features. We addressed the multi-modal data challenge by proposing a family of oversamplers based on the principle of encoding the data in a dense continuous latent space, interpolate there and map the samples back to the original feature space. We experimented with real and artificial datasets using several autoencoder schemes, two underlying interpolation methods and several other previously proposed oversamplers. Our approach yields a superior prediction quality for real-world datasets and generates better synthetic data compared to all other methods.

We are interested in further exploring the proposed framework in several directions: As we currently train our autoencoders in an unsupervised fashion, we plan to study how the labels can be incorporated in the autoencoder training phase to encourage better interpolation. We also plan to investigate how can our framework be used to benefit other problem types e.g., "balanced" classification or regression. Additionally, modern tabular datasets might include data types such as free text, images, and audio. Support for synthesising of multi-modal data that includes these data types is an interesting avenue for future work.

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[3] N. Kalchbrenner and P. Blunsom, “Recurrent continuous translation models,” in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1700–1709.

[4] L. Schmarje, M. Santarossa, S. Schröder, and R. Koch, “A survey on semi-, self- and unsupervised techniques in image classification,” *CoRR*, vol. abs/2002.08721, 2020. [Online]. Available: https://arxiv.org/abs/2002.08721

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.

[6] D. Teres, S. Lemeshow, J. S. Avrunin, and H. Pastides, “Validation of the mortality prediction model for icu patients.” *Critical care medicine*, vol. 15, no. 3, pp. 208–213, 1987.

[7] A. J. Campbell, J. A. Cook, G. Adey, and B. H. Cuthbertson, “Predicting death and readmission after intensive care discharge,” *British journal of anaesthesia*, vol. 100, no. 5, pp. 656–662, 2008.

[8] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in Neural Information Processing Systems 26*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2013, pp. 3111–3119.

[10] X. Cai, J. Gao, K. Y. Ngiam, B. C. Ooi, Y. Zhang, and X. Yuan, “Medical concept embedding with time-aware attention,” *CoRR*, vol. abs/1806.02873, 2018. [Online]. Available: http://arxiv.org/abs/1806.02873

[11] ——, “Medical concept embedding with time-aware attention,” *arXiv preprint arXiv:1806.02873*, 2018.

[12] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[13] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[14] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[15] D. E. Rumelhart, G. E. Hinton, R. J. Williams *et al.*, "Learning representations by back-propagating errors," *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[16] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[17] E. Choi, M. T. Bahadori, E. Searles, C. Coffey, M. Thompson, J. Bost, J. Tejedor-Sojo, and J. Sun, "Multi-layer representation learning for medical concepts," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM, 2016, pp. 1495–1504.

[18] P. Nguyen, T. Tran, N. Wickramasinghe, and S. Venkatesh, "`Deepr`: a convolutional net for medical records," *IEEE journal of biomedical and health informatics*, vol. 21, no. 1, pp. 22–30, 2016.

[19] E. Choi, M. T. Bahadori, L. Song, W. F. Stewart, and J. Sun, "Gram: Graph-based attention model for healthcare representation learning," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 787–795. [Online]. Available: http://doi.acm.org/10.1145/3097983.3098126

[20] J. Zhang, K. Kowsari, J. H. Harrison, J. M. Lobo, and L. E. Barnes, "Patient2vec: A personalized interpretable deep representation of the longitudinal electronic health record," *IEEE Access*, vol. 6, pp. 65 333–65 346, 2018.

[21] E. Choi, C. Xiao, W. Stewart, and J. Sun, "Mime: Multilevel medical embedding of electronic health records for predictive healthcare," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 4547–4557.

[22] D. Galvan, N. Okazaki, K. Matsuda, and K. Inui, "Investigating the challenges of temporal relation extraction from clinical text," in *Proceedings of the Ninth International Workshop on Health Text Mining and Information Analysis.* Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 55–64. [Online]. Available: https://www.aclweb.org/anthology/W18-5607

[23] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun *et al.*, "Scalable and accurate deep learning with electronic health records," *NPJ Digital Medicine*, vol. 1, no. 1, p. 18, 2018.

[24] D. Dligach and T. A. Miller, "Learning patient representations from text," *CoRR*, vol. abs/1805.02096, 2018. [Online]. Available: http://arxiv.org/abs/1805.02096

[25] J. Liu, Z. Zhang, and N. Razavian, "Deep ehr: Chronic disease prediction using medical notes," *arXiv preprint arXiv:1808.04928*, 2018.

[26] K. Huang, J. Altosaar, and R. Ranganath, "Clinicalbert: Modeling clinical notes and predicting hospital readmission," *arXiv:1904.05342*, 2019.

[27] T. Bai, A. K. Chanda, B. L. Egleston, and S. Vucetic, "Ehr phenotyping via jointly embedding medical concepts and words into a unified vector space," *BMC medical informatics and decision making*, vol. 18, no. 4, p. 123, 2018.

[28] J. Mullenbach, S. Wiegreffe, J. Duke, J. Sun, and J. Eisenstein, "Explainable prediction of medical codes from clinical text," *arXiv preprint arXiv:1802.05695*, 2018.

[29] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning.* ACM, 2008, pp. 1096–1103.

[31] M. Kachuee, S. Darabi, B. Moatamed, and M. Sarrafzadeh, "Dynamic feature acquisition using denoising autoencoders," *CoRR*, vol. abs/1811.01249, 2018. [Online]. Available: http://arxiv.org/abs/1811.01249

[32] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "Biobert: pre-trained biomedical language representation model for biomedical text mining," *arXiv preprint arXiv:1901.08746*, 2019.

[33] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific data*, vol. 3, p. 160035, 2016.

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] E. Alsentzer, J. R. Murphy, W. Boag, W.-H. Weng, D. Jin, T. Naumann, and M. McDermott, "Publicly available clinical bert embeddings," *arXiv preprint arXiv:1904.03323*, 2019.

[36] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[37] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," in *NIPS Autodiff Workshop*, 2017.

[38] P. Baldi, "Autoencoders, unsupervised learning, and deep architectures," in *Proceedings of ICML workshop on unsupervised and transfer learning*, 2012, pp. 37–49.

[39] S. Darabi, M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "Taper: Time-aware patient ehr representation," *arXiv preprint arXiv:1908.03971*, 2019.

[40] P. Szolovits and S. G. Pauker, "Categorical and probabilistic reasoning in medical diagnosis," *Artificial Intelligence*, vol. 11, no. 1-2, pp. 115–144, 1978.

[41] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[42] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[43] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber, "Stacked convolutional auto-encoders for hierarchical feature extraction," in *International Conference on Artificial Neural Networks.* Springer, 2011, pp. 52–59.

[44] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[46] R. Miotto, L. Li, B. A. Kidd, and J. T. Dudley, "Deep patient: an unsupervised representation to predict the future of patients from the electronic health records," *Scientific reports*, vol. 6, p. 26094, 2016.

[47] H. Suresh, P. Szolovits, and M. Ghassemi, "The use of autoencoders for discovering patient phenotypes," *arXiv preprint arXiv:1703.07004*, 2017.

[48] E. Choi, Z. Xu, Y. Li, M. W. Dusenberry, G. Flores, Y. Xue, and A. M. Dai, "Graph convolutional transformer: Learning the graphical structure of electronic health records," 2019.

[49] M. Kachuee, S. Fazeli, and M. Sarrafzadeh, "Ecg heartbeat classification: A deep transferable representation," in *2018 IEEE International Conference on Healthcare Informatics (ICHI).* IEEE, 2018, pp. 443–444.

[50] E. P. Lehman, R. G. Krishnan, X. Zhao, R. G. Mark, and L. H. Lehman, "Representation learning approaches to detect false arrhythmia alarms from ecg waveforms."

[51] X. Lyu, M. Hüser, S. L. Hyland, G. Zerveas, and G. Rätsch, "Improving clinical predictions through unsupervised time series representation learning," *arXiv preprint arXiv:1812.00490*, 2018.

[52] T. J. Pollard, A. E. Johnson, J. D. Raffa, L. A. Celi, R. G. Mark, and O. Badawi, "The eicu collaborative research database, a freely available multi-center database for critical care research," *Scientific data*, vol. 5, 2018.

[53] J. E. Zimmerman, A. A. Kramer, D. S. McNair, and F. M. Malila, “Acute physiology and chronic health evaluation (apache) iv: hospital mortality assessment for today’s critically ill patients,” *Critical care medicine*, vol. 34, no. 5, pp. 1297–1310, 2006.

[54] J. Howard and S. Ruder, “Universal language model fine-tuning for text classification,” *arXiv preprint arXiv:1801.06146*, 2018.

[55] B. Felbo, A. Mislove, A. Søgaard, I. Rahwan, and S. Lehmann, “Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm,” *arXiv preprint arXiv:1708.00524*, 2017.

[56] C. S. Crowson, E. J. Atkinson, and T. M. Therneau, “Assessing calibration of prognostic risk scores,” *Statistical methods in medical research*, vol. 25, no. 4, pp. 1692–1706, 2016.

[57] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, “Calibrating predictive model estimates to support personalized medicine,” *Journal of the American Medical Informatics Association*, vol. 19, no. 2, pp. 263–274, 2011.

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

[59] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations,” 2020.

[60] R. Zhang, P. Isola, and A. A. Efros, “Colorful image colorization,” in *European conference on computer vision*. Springer, 2016, pp. 649–666.

[61] M. Sajjadi, M. Javanmardi, and T. Tasdizen, “Regularization with stochastic transformations and perturbations for deep semi-supervised learning,” *arXiv preprint arXiv:1606.04586*, 2016.

[62] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *arXiv preprint arXiv:1803.07728*, 2018.

[63] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *arXiv preprint arXiv:1310.4546*, 2013.

[64] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “mixup: Beyond empirical risk minimization,” *arXiv preprint arXiv:1710.09412*, 2017.

[65] A. Iscen, G. Tolias, Y. Avrithis, and O. Chum, “Label propagation for deep semi-supervised learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5070–5079.

[66] D. hyun Lee, “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks.”

[67] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision.* Springer, 2016, pp. 69–84.

[68] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[69] A. Mnih and G. E. Hinton, "A scalable hierarchical distributed language model," *Advances in neural information processing systems*, vol. 21, pp. 1081–1088, 2008.

[70] L. Logeswaran and H. Lee, "An efficient framework for learning sentence representations," *arXiv preprint arXiv:1803.02893*, 2018.

[71] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[72] P. Yin, G. Neubig, W.-t. Yih, and S. Riedel, "Tabert: Pretraining for joint understanding of textual and tabular data," *arXiv preprint arXiv:2005.08314*, 2020.

[73] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," *arXiv preprint arXiv:1908.07442*, 2019.

[74] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 2016, pp. 1857–1865.

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

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

[77] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," *arXiv preprint arXiv:1703.01780*, 2017.

[78] Y. Grandvalet, Y. Bengio *et al.*, "Semi-supervised learning by entropy minimization." in *CAP*, 2005, pp. 281–296.

[79] J. Yoon, Y. Zhang, J. Jordon, and M. van der Schaar, "Vime: Extending the success of self-and semi-supervised learning to tabular domain," *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[80] M. Douze, A. Szlam, B. Hariharan, and H. Jégou, "Low-shot learning with large-scale diffusion," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3349–3358.

[81] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[82] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *International Conference on Machine Learning.* PMLR, 2019, pp. 6438–6447.

[83] X. Zhu, J. Lafferty, and R. Rosenfeld, "Semi-supervised learning with graphs," Ph.D. dissertation, Carnegie Mellon University, language technologies institute, school of . . . , 2005.

[84] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," *arXiv preprint arXiv:1706.09516*, 2017.

[85] N. V. Chawla, "Smote : Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002. [Online]. Available: https://ci.nii.ac.jp/naid/20001554764/en/

[86] R. Blagus and L. Lusa, "Evaluation of smote for high-dimensional class-imbalanced microarray data," in *2012 11th International Conference on Machine Learning and Applications*, vol. 2, 2012, pp. 89–94.

[87] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Applied Soft Computing*, vol. 83, p. 105662, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1568494619304429

[88] L. Prokhorenkova, G. Gusev, A. Vorobev, A. V. Dorogush, and A. Gulin, "Catboost: unbiased boosting with categorical features," in *Advances in neural information processing systems*, 2018, pp. 6638–6648.

[89] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu, "Lightgbm: A highly efficient gradient boosting decision tree," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 3146–3154. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf

[90] S. O. Arik and T. Pfister, "Tabnet: Attentive interpretable tabular learning," 2020.

[91] A. Fernández, S. García, F. Herrera, and N. V. Chawla, "Smote for learning from imbalanced data: Progress and challenges, marking the 15-year anniversary," *J. Artif. Int. Res.*, vol. 61, no. 1, p. 863–905, Jan. 2018.

[92] S. Gazzah and N. E. B. Amara, "New oversampling approaches based on polynomial fitting for imbalanced data sets," in *2008 The Eighth IAPR International Workshop on Document Analysis Systems*, 2008, pp. 677–684.

[93] S. Barua, M. M. Islam, and K. Murase, “Prowsyn: Proximity weighted synthetic oversampling technique for imbalanced data set learning,” in *Advances in Knowledge Discovery and Data Mining*, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 317–328.

[94] J. Wang, M. Xu, H. Wang, and J. Zhang, “Classification of imbalanced data by using the smote algorithm and locally linear embedding,” in *2006 8th international Conference on Signal Processing*, vol. 3, 2006.

[95] Q. Gu, Z. Cai, and L. Zhu, “Classification of Imbalanced Data Sets by Using the Hybrid Re-sampling Algorithm Based on Isomap,” in *Advances in Computation and Intelligence*, Z. Cai, Z. Li, Z. Kang, and Y. Liu, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 287–296.

[96] M. Pérez-Ortiz, P. A. Gutiérrez, P. Tino, and C. Hervás-Martínez, “Oversampling the minority class in the feature space,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 9, pp. 1947–1961, 2016.

[97] B. Tang and H. He, “Kerneladasyn: Kernel based adaptive synthetic data generation for imbalanced learning,” in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015, pp. 664–671.

[98] J. Mathew, M. Luo, C. K. Pang, and H. L. Chan, “Kernel-based smote for svm classification of imbalanced datasets,” in *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, 2015, pp. 001 127–001 132.

[99] C. Bellinger, N. Japkowicz, and C. Drummond, “Synthetic oversampling for advanced radioactive threat detection,” in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, 2015, pp. 948–953.

[100] C. Bellinger, C. Drummond, and N. Japkowicz, “Beyond the boundaries of smote - a framework for manifold-based synthetically oversampling,” in *ECML/PKDD*, 2016.

[101] K. Babaei, Z.-Y. Chen, and T. Maul, “Data augmentation by autoencoders for unsupervised anomaly detection,” *ArXiv*, vol. abs/1912.13384, 2019.

[102] S. K. Lim, Y. Loo, N. Tran, N. Cheung, G. Roig, and Y. Elovici, “Doping: Generative data augmentation for unsupervised anomaly detection with gan,” in *2018 IEEE International Conference on Data Mining (ICDM)*, 2018, pp. 1122–1127.

[103] Z. Xie, L. Jiang, T. Ye, and X. Li, “A synthetic minority oversampling method based on local densities in low-dimensional space for imbalanced learning,” in *Database Systems for Advanced Applications*, M. Renz, C. Shahabi, X. Zhou, and M. A. Cheema, Eds. Cham: Springer International Publishing, 2015, pp. 3–18.

[104] S. Tang and S. Chen, “The generation mechanism of synthetic minority class examples,” in *2008 International Conference on Information Technology and Applications in Biomedicine*, 2008, pp. 444–447.

[105] H. Zhang, M. Cissé, Y. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *ArXiv*, vol. abs/1710.09412, 2018.

[106] X. Yin, X. Yu, K. Sohn, X. Liu, and M. Chandraker, "Feature transfer learning for deep face recognition with under-represented data," 2019.

[107] S. Carter and M. Nielsen, "Using artificial intelligence to augment human intelligence," *Distill*, 2017, https://distill.pub/2017/aia.

[108] D. Berthelot, C. Raffel, A. Roy, and I. Goodfellow, "Understanding and improving interpolation in autoencoders via an adversarial regularizer," 2018.

[109] L. Xu and K. Veeramachaneni, "Synthesizing tabular data using generative adversarial networks," 2018.

[110] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional gan," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 7335–7345. [Online]. Available: http://papers.nips.cc/paper/8953-modeling-tabular-data-using-conditional-gan.pdf

[111] N. Park, M. Mohammadi, K. Gorde, S. Jajodia, H. Park, and Y. Kim, "Data synthesis based on generative adversarial networks," *Proc. VLDB Endow.*, vol. 11, no. 10, p. 1071–1083, Jun. 2018. [Online]. Available: https://doi.org/10.14778/3231751.3231757

[112] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," ser. Proceedings of Machine Learning Research, F. Doshi-Velez, J. Fackler, D. Kale, R. Ranganath, B. Wallace, and J. Wiens, Eds., vol. 68. Boston, Massachusetts: PMLR, 18–19 Aug 2017, pp. 286–305. [Online]. Available: http://proceedings.mlr.press/v68/choi17a.html

[113] R. Fakoor, J. Mueller, N. Erickson, P. Chaudhari, and A. J. Smola, "Fast, accurate, and simple models for tabular data via augmented distillation," 2020.

[114] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.

[115] P. Ghosh, M. S. M. Sajjadi, A. Vergari, M. J. Black, and B. Schölkopf, "From variational to deterministic autoencoders," *CoRR*, vol. abs/1903.12436, 2019. [Online]. Available: http://arxiv.org/abs/1903.12436

[116] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein gan," 2017.

[117] Z. Ding, "Diversified ensemble classifiers for highly imbalanced data learning and its application in bioinformatics," Ph.D. dissertation, USA, 2011.