6-18-2022

Generalized Representation of Electronic Health Records for Unplanned Hospital Readmission

Sanjay Krishnamurthy
Indian Institute of Technology, Madras, t.k.sanjay@gmail.com

Nargis Pervin
IIT Madras, nargisp@iitm.ac.in

GENERALIZED REPRESENTATION OF ELECTRONIC HEALTH RECORDS FOR UNPLANNED HOSPITAL READMISSION

Research Paper

Sanjay K, Indian Institute of Technology Madras, India, t.k.sanjay@gmail.com
Nargis Pervin, Indian Institute of Technology Madras, India, nargisp@iitm.ac.in

Abstract

*Unplanned hospital readmissions soon after a person is discharged indicate the poor performance of the healthcare. Previous attempts of readmission prediction pose it as a binary classification problem and largely ignore the previous history. This study proposes a novel neural network architecture called **S**equential **R**eadmission **P**redictor with **M**ultitasking (SRPM), to enhance the existing readmission prediction models. We retain the previous admission history of a patient by learning a latent representation for the patient, which could be used for every new admission by the same patient. Our proposed model uses a multitask neural network model that simultaneously models it as a binary classification problem and as a regression problem that predicts the exact days of readmission. By doing so, the error information from the regression task augments the classification task. The results show a promising improvement of up to 6.59% in AUCROC and 19% in F1 score over four benchmark methods.*

Keywords: Electronic Health Record, hospital readmission, healthcare, multitask recommender.

1 Introduction

Advances in healthcare information systems have increased over the years worldwide. They help improve the quality and efficiency of the healthcare system, enhance interactions between patients and providers, and enable greater access to the latest advancements in treatments. A major component of healthcare information systems is the Electronic Health Record (EHR). As per U.S. Department of Health and Human Services, EHRs are real-time, patient-centered records that make information available instantly and securely to authorized users. EHRs can contain a patient's medical history, diagnoses, medications, treatment plans, immunization dates, allergies, radiology images, and laboratory and test results. The recent shifts in healthcare policy such as The ACA have recommended health practices to focus on preventive care to improve the overall health of the population (Wager, Lee, and Glaser, 2021). The work by Hillestad et al. (2005) suggests that widespread adoption of EHR could lead to many health and safety benefits and

could bring down the global healthcare costs by $81 billion annually. Evaluating risk assessment of the patient, finding disease correlations, drug interactions, etc. P. B. Jensen, L. J. Jensen, and Brunak (2012) are examples of important use cases where machine learning algorithms were utilized on EHR data.

In this context, deep learning techniques are well suited to analyze high dimensional data that are sequential in nature (e.g., text data, as order of words in sentences are important to determine context), and the sequences are of variable length. Specifically, this proves to be extremely useful to build models on EHR data where for every patient, there are varying lengths of medical records like the list of diagnosis, medical procedures, and the drugs prescribed for the patient during their stay. The pioneering work by Rajkomar et al. (2018) employs a deep learning technique to predict inpatient mortality, readmissions, length of stay in the hospital, and discharge diagnosis.

A typical EHR record for a patient spans multiple tables and each EHR record is characterized by high-dimensionality (14,000 diagnostic codes by International classification of diseases (ICD-9)), temporality (variable length), and sparsity (they span across multiple tables and not every patient has an entry in all the table) Cheng et al. (2016). Modeling such irregular high dimensional data poses a significant challenge using popular modeling choices like XGBoost and LightGBM. These methods ignore all the sequential information and consider only the patient's admission level statistic to predict the outcome. To tackle this issue, we propose a deep learning architecture, where all the tables linked to each record of a patient's admission are represented as an embedding and fed to a final model that predicts the readmission probabilities.

2 Related work

Electronic health records are instrumental in enhancing patient care through health information technology (Manga and J. Sun, 2020). The role of EHR in big data analytics is addressed in several contexts, including that of clinical decision-making (Amarasingham et al., 2014; Wickramasinghe, Moghimi, and Schaffer, 2017) and medication management (Hernandez and Yuting Zhang, 2017), to predict diseases (Manias et al., 2018), specific applications in mental health (Hahn, Nierenberg, and Whitfield-Gabrieli, 2017) and precision public health (Khoury et al., 2018). However, since our objective is to predict the unplanned readmission probability, we have restricted the literature to risk assessment, disease prediction, length of stay prediction, and unplanned readmission using EHR data. The first work using neural networks to predict disease called Deep patient (Miotto et al., 2016) arrive at a patient representation using a three-layer stacked autoencoder. Each encoder finds a latent representation for the patient's medications, diagnosis, and procedures. The patient representation is then used as input to a random forest classifier, predicting the probability of developing a future disease. Since this approach uses autoencoders that are unsupervised models, the representation is more generic and is not focused on a particular task. Random forests to predict the readmission probability are used by Deschepper et al. (2019) and Wong et al. (2021) where they find the variable importance of the features. They have shown that diagnosis, drugs, and procedures are the crucial factors in predicting readmission probability. While random forests are better explainable than neural networks, their predictive ability is restricted to tabular data. A CNN-based technique is presented in Cheng et al. (2016), where each patient is represented as a longitudinal event matrix having time in the x-axis and ICD9 code for diagnosis in the y-axis.

Three convolutional kernels are slid on top of each other, followed by a final fully connected layer to arrive at the vector representation used for predictive modeling. This work, however, does not consider the time difference between the events, which is essential to know if the events have any relation in common. This is addressed in the paper by DeepR (Nguyen et al., 2017), where each visit of a patient to a hospital is made of a set of ICD9 codes for diagnosis and procedures. The ICD9 codes are then converted to embedding using Word2Vec. Thus, each visit is made of a stack of vectors, and multiple such visits make a medical record. The time difference between the visits is also provided as a vector. To learn a meaningful representation of the medical record, a 1D CNN is slid on the stack, and the final vector obtained through max-pooling is used to classify readmission. This representation, however, lacks patient-specific attributes like age, gender, etc. To overcome this, an extension to DeepR was done by (Balan U, Gandhi, Rammohan, et al., 2021) where static features of the patient like age, sex, ethnicity, etc. are given as input to the last layer of the LSTM or CNN on top of sequential features like diagnosis and procedures data. Although static features are critical, they do not have a sequential relationship between themselves and therefore, feeding it as the last layer to an LSTM that takes sequential elements may not give a maximum improvement in performance. To overcome this, we posit a model where the output of the sequential representation using GRUs is concatenated with the static feature representation would be beneficial. Multitask learning has been successfully applied in mortality prediction Si and Roberts (2019) and Yu et al. (2019), implying that learning multiple tasks is better than learning a single task. While Si and Roberts (2019) has the auxiliary task of predicting mortality at different periods (after six months, one year, etc.), Yu et al. (2019) uses the auxiliary task as predicting a sequence of critical physiological measurements within the first 24 hours of admission. The secondary tasks in both works are not very closely related to the primary tasks, which may not result in maximum improvement in primary task performance. We address this by having the secondary task as predicting the exact days of readmission after discharge, which complements the primary task of readmission prediction within 30 days.

3 Dataset Description and preprocessing

We used the freely available de-identified health-related dataset MIMIC III, which contains the clinical data of 58,976 admissions of 46,520 patients admitted to the Beth Israel Deaconess Medical Center in Boston, Massachusetts, between 2001 and 2012. Out of all the patients, 7537 patients had been readmitted more than once, making the average readmission rate 1.27 per patient. Since only 16.2% of the patients were readmitted, highlighting that the dataset is imbalanced towards the task at hand, we bootstrapped the records of readmitted patients to match those who were not readmitted. The MIMIC dataset is normalized across multiple tables, and we used four important tables: the admissions, diagnosis, procedures, and prescriptions. The admissions table is the main table that contains patients' features like gender, age, date of admission, and discharge. As each feature can take a single value for a patient admission, we call them static features. The diagnosis, procedures, and prescriptions tables contain the sequence of diagnosis, procedures, and prescriptions undergone by the patient admission, and so we call them sequential features. The subsections below go through the preprocessing steps for each of them.

Static Features	**Description**
Subject ID	Identifier for a patient
Admission ID	Identifier for an admission
Age	Age of the patient (D)
Duration	Difference between the time when patient was admitted and discharged in hours (D)
Total Duration	Sum of all previous admissions in hours (D)
Previous admissions	Count of all previous admissions (D)
Admission type	Describes the type of the admission: 'Elective', 'Urgent', 'Newborn' or 'Emergency'
Admission location	Previous location of the patient prior to arriving at the hospital. It can take one of 9 possible values
Discharge location	Location at which patient is discharged. It can take one of 17 possible values
Insurance	It can take one of 5 possible insurance options
Language	Native language of the patient. It can take one of 75 possible values
Marital status	It can take one of 7 possible values.
Ethnicity	It can take one of 41 possible values
ICU duration	Difference between the time when patient was admitted in ICU and discharged from ICU in hours (D)
Disease	Indicates the type of disease. It can take one of 15691 possible values.
Expire flag	Indicator for patient died
Sequential Features	
Prescriptions	National Drug Code for the drugs sequence prescribed to the patient.
Diagnosis	ICD9 code for diagnosis sequence
Proceedures	ICD9 code for Proceedures sequence

(D) - Derived features

Table 1. *Static and Sequential features of a patient*

3.1. Static features

Each record of the admissions table contains a unique admission ID used to connect with the other three tables. From the static features in the admission table, we did feature engineering to arrive at derived features, which are listed in Table 1. Since the disease column that lists the type of disease has a cardinality of 15691, it is encoded using both one-hot encoding to get a sparse representation and an embedding layer to get a dense representation. The sparse representation is used to memorize certain combinations whereas the dense representation is used for generalization. All other categorical variables have low cardinality and are one-hot encoded, and numerical variables are standardized.

Sample Input						
Patient_id	***Admission_id***	***Duration***	***Total duration***	***Prescriptions***	***Diagnosis***	***Proceedures***
55357	119355	857	518	641039425	V3401	9390
				63323038810	769	9915
Previous admissions	***Admission type***	***Admission_location***		338002304	9992	966
2	Urgent	Phys referral		517293025	7742	9983
				74978901	7793	9955
Discharge location	***Insurance***	***Language***	***Marital_status***	63323022110	76518	
Home health care	Private	English	Married	641039425	77081	
				87036503	V502	
Ethnicity	***ICU duration***	***Disease***	***Expire_flag***	87040303	V290	
Black/African American	52	Laryngeal Edema	0		V053	
					6910	
Sample Output						
Readmission Probability		***Exact days of readmission***				
0.15		92				

Table 2. *Sample input and output*

Statistic	**Value**
# Mean diagnosis per admission	11.03
# Mean procedures per admission	4.59
# Mean drugs administered per patient admission	82.77
# Unique ICD9 codes for Diagnosis	6984
# Unique ICD9 codes for Procedures	2009
# Unique Drugs	4525

Table 3. *Statistics of sequential features*

Figure 1. *Variable length sequences are padded with zeros to the left to get a fixed-length representation*

3.2. Sequential features

The sequential features of the patient admission are contained in the diagnosis, procedures, and prescriptions table. The diagnosis table has the sequence of diagnoses undergone by a patient identified by ICD9 codes. The prescriptions table contains the list of drugs administered to the patient while admitted to the hospital. Finally, the procedures table contains the procedures done on the patient coded by ICD9 code for procedures. The list of sequential features used in the

experiment is shown in Table 1, and their basic statistics are listed in Table 3. Each admission ID has an associated sequence of records in the diagnosis, procedures, and prescriptions tables, and a sample of the input features and associated output are shown in Table 2. To map the sequence of features to a single admission ID, we group them by admission ID and pad it with zeros to the left such that the sequences are of fixed length 'T'. Any sequence which is greater than the 'T' is truncated on the left such that the recent events are on the right, and shorter length sequences are zero-padded to the left as shown in Figure 1. The features are tokenized, and if a particular token appears in the test set but is not present in the train set, it is treated as out of vocabulary token and handled appropriately.

4 Model Architecture

The architecture consists of a memory augmented Gated Recurrent Unit (GRU) for modeling sequential features and a feed-forward network for modeling static features and hence called sequential readmission predictor. The overall model architecture is shown in Figure 2. The sequential and static features of the patient are then concatenated and passed through a feed-forward neural network. It is then passed through two towers of feed-forward networks. The tower on the left (Task A) is used for binary classification, which predicts the probability of readmission, while the tower on the right (Task B) predicts the exact days of readmission. Since both towers share a common layer, the error gradients from Task B will adjust the weights in the common layer to complement task A. The details of the architecture are described below.

4.1. Embedding Layer

The embedding layer is a lookup table that takes the category number as input and produces a vector representation of the category. After converting the data in each of the diagnosis, prescriptions and procedures table into fixed-length sequences 'T', it is fed to an embedding layer to get the latent vector representation $\{\mathbf{e}_1, \ldots \mathbf{e}_t, \ldots . \mathbf{e}_T\}$ where 't' is the timestep and $\mathbf{e}_t \in \mathbb{R}^{d \times 1}$. The final embedding matrix after passing through the embedding layer can be given as,

$$\mathbf{E} = \begin{bmatrix} \mathbf{e}_1 \\ \vdots \\ \mathbf{e}_t \\ \vdots \\ \mathbf{e}_T \end{bmatrix} \in \mathbb{R}^{d \times n}$$

In this way we obtain an embedding representation for each patient $\mathbf{e}_p \in \mathbb{R}^{d_1}$, drugs $\mathbf{e}_d \in \mathbb{R}^{d_2}$ procedures $\mathbf{e}_t \in \mathbb{R}^{d_3}$ and diagnosis $\mathbf{e}_s \in \mathbb{R}^{d_4}$ where d_1, d_2, d_3, d_4 are the embedding dimensions and is set to $\text{vocab}^{0.25}$ where *vocab* is the cardinality of the categorical variable.

4.2. Gated Recurrent unit (GRU)

GRU is an advanced variant of neural network that is helpful in modeling sequence data. The embedding matrix output from the previous layer is fed as input to the GRU which produces a

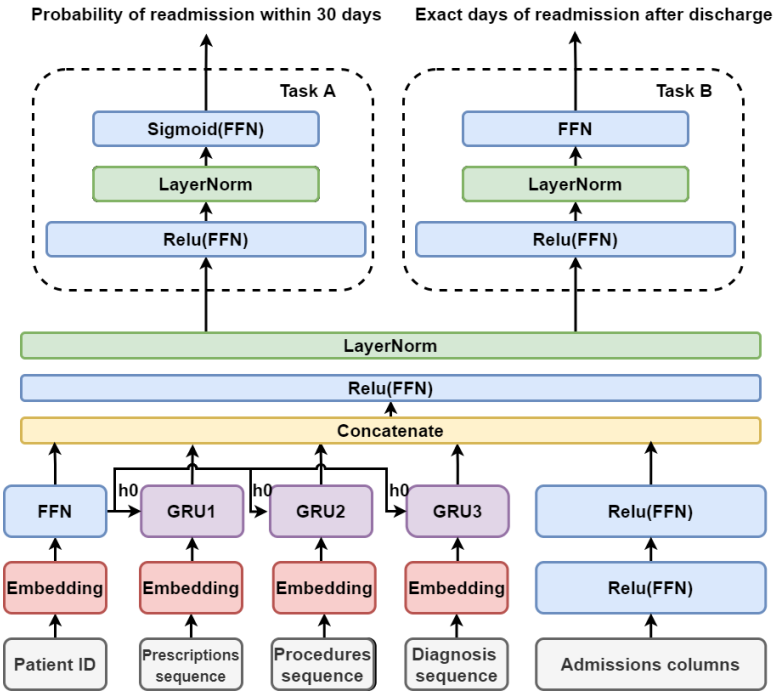

Figure 2. *Architecture of our proposed model SRPM. The GRUs are patient personalized by initializing it with a linear transformation of the patient embeddings.*

vector output at each timestep. Formally, at each timestep, the GRU takes the embedding vector $\mathbf{e}_t$, the previous hidden state vector $\mathbf{h}_{t-1}$ and computes the current hidden state $\mathbf{h}_t$ given by

$$\mathbf{h}_t = GRU(\mathbf{e}_t, \mathbf{h}_{t-1}), \quad t = 1, \ldots, T$$

The initial hidden state $\mathbf{h_0}$ of the GRU is initialized to a linear transformation of the patient embedding

$$\mathbf{h}_0 = \mathbf{W}_0 \mathbf{e}_p$$

This way, the GRU can have prior knowledge about the patients' historical data rather than having a random vector as the initial state. We found that having such an initialization improved the overall AUCROC by 7%. The final output of the GRU is

$$\mathbf{H} = Concat(\mathbf{h}_0, \ldots, \mathbf{h}_T)$$

The embedding output from the drugs, procedures, and diagnosis is fed as input to the three GRUs, and the outputs obtained are $\mathbf{H}_1, \mathbf{H}_2$ and $\mathbf{H}_3$

4.3. Multitask Learning

The input static features $\mathbf{X}^s$ is sent through a series of feed forward neural network (FFN) with ReLU activation to obtain $\mathbf{H}^s$. The disease feature in $\mathbf{X}^s$ is represented both as one-hot encoding

and as an embedding $\mathbf{e}_x$.

$$\mathbf{H}^s = ReLU\left(\left(\mathbf{X}^s\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2\right)$$
$$\mathbf{H} = Concat\left(\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3, \mathbf{H}^s, \mathbf{e}_x\right)$$
$$\mathbf{H} = LayerNorm(ReLU\left(\mathbf{H}\mathbf{W}_3\right) + \mathbf{b}_3)$$

Multitask learning is proven beneficial when the final prediction tasks have a lot of similarities with an auxiliary task. This is because learning to optimize for both the tasks jointly will help share parameters that complement each other. Since readmission on the 3rd day and the 28th day are the same when considering readmission probability within 30 days, there is a loss of information that could be complemented by an auxiliary task B, which is to learn the exact days of readmission. To implement this, the output from the GRUs and $\mathbf{H}^s$ are then concatenated and is then fed as input for two tasks which are Task A and Task B. The Task A tower consists of a stack of FFNs with LayerNorm, and the Task B tower also consists of such a stack. Both the towers share a common FFN network at the bottom layer. The task-specific parameters are learned in the separate tower parameters. The output $\mathbf{O}_1$ after passing through tower A is given as,

$$\mathbf{o}_1 = ReLU(FFN(\mathbf{H}))$$
$$\hat{\mathbf{y}^1} = Sigmoid(FFN(LayerNorm(\mathbf{o}_1)))$$

Similarly for the secondary task B, the output is,

$$\mathbf{o}_2 = ReLU(FFN(\mathbf{H}))$$
$$\hat{\mathbf{y}^2} = FFN(LayerNorm(\mathbf{o}_2))$$

4.4. Model Prediction and Loss function

We use two loss functions for the two tasks of the model to learn the weights from training. For task A, we use the **B**inary **C**ross-**E**ntropy loss (BCE Loss) as the optimization objective to predict the readmission probability within 30 days. BCE Loss takes the output probability $\hat{y}_i$ and the actual label y_i, which is either 0 or 1 and produces an output depending on how close the prediction is to the actual label. It is given by,

$$\text{BCE Loss} = -\frac{1}{N}\sum_{i=1}^{N} y_i log(\hat{y}_i) + (1-y_i) log(1-\hat{y}_i)$$

The output from the auxiliary task B is trained using **M**ean **S**quared **L**oss (MSE Loss) and the loss function given by,

$$\text{MSE Loss} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

5 Experiment Details

5.1. Baselines

We compare our model Sequential Readmission Predictor with Multitasking (SRPM) with the baselines briefly described below,

- **Logistic Regression** applies logistic function to model the static variables.
- **DeepR** (Nguyen et al., 2017) encodes visits comprised of ICD9 codes of diagnosis and procedures using word2vec and applies a CNN on top to classify.
- **Hybrid DeepR** (Balan U, Gandhi, Rammohan, et al., 2021) is an extension to DeepR where static features are used in the last layer.
- **PURE** (Balan U, Gandhi, Rammohan, et al., 2021) uses Bidirectional LSTM to model the sequential features

5.2. Experimental Setup

Python version 3.7.3 has been used on a Windows system with a 1.99 GHz Intel Core i7 processor and 32GB RAM for the experiments. We used TensorFlow version 2.3.1 to implement our model and scikit-learn (Pedregosa et al., 2011) for preprocessing.

Model	AUC	Accuracy	Sensitivity	Specificity	Precision	F1	F1 Improvement
Logistic	0.72	0.71	0.98	0.2	0.2	0.33	NA
DeepR	0.86	0.83	0.94	0.62	0.62	0.75	127%
Hybrid DeepR	0.88	0.86	0.97	0.64	0.64	0.77	3%
PURE	0.91	0.87	0.97	0.67	0.67	0.79	3%
SRPM	0.97	0.96	1	0.88	0.88	0.94	19%

Table 4. *Performance of baseline methods across metrics*

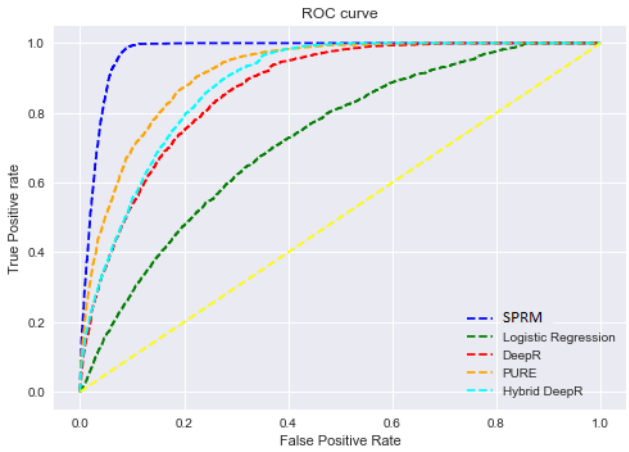

Figure 3. *AUCROC plot for all the models.*

	TRP of Models				
FPR	Logistic	DeepR	Hybrid DeepR	PURE	SRPM
0.1	0.283	0.533	0.541	0.684	0.968
0.2	0.457	0.744	0.782	0.856	0.982
0.3	0.6	0.868	0.907	0.941	0.984
0.4	0.704	0.936	0.976	0.977	0.986
0.5	0.799	0.966	0.986	0.988	0.991
0.6	0.881	0.979	0.993	0.993	0.996
0.7	0.927	0.986	1	1	1
0.8	0.969	1	1	1	1
0.9	0.996	1	1	1	1
1	1	1	1	1	1

Table 5. *Comparison of AUCROC values for the different models*

The early stopping criteria have been considered for all experiments, and the patience parameter was set to 4. Adam optimizer was used with a learning rate of 0.001 and batch size 16, which resulted in the best performance. The model hyperparameters are tuned based on the validation set via randomized search, and the learning rate was set using the learning rate scheduler. Regularization techniques like Layer normalization and dropout were also applied to reduce overfitting and improve test performance. The data is split into train and test in the ratio 80:20, and the 20% of the train set is used as the validation set. The model results are averaged over ten runs, and in each run, a different seed value is set to ensure generality. AUCROC score, accuracy, sensitivity, specificity, precision, and F1 metrics are used to evaluate our model's efficacy, and the results are reported in the Table 4. The AUCROC plot is shown in Figure 3 and the actual values are reported in Table 5. We have used the F1 score along with the AUCROC metric for the target of readmission prediction.

6 Ablation study

This section describes how individual components of the AI system contribute to the overall performance of the system. We do this by introducing a series of model increments A to F as listed in Table 6 to arrive at the final model. Adding one increment on top of another resulted in better accuracy, and there is no particular meaning to the order in which we have listed. We found that using the static features alone (A) gave an improved performance, as reported in Table 7. Since the data is imbalanced, adding bootstrapping to it (A+B) gave a significant boost in performance. We found that bootstrapping the minority class to 10 times the size resulted in maximum improvement.

On top of the static features and bootstrapping, adding the sequential features (A+B+C) using the GRU outputs at all the timesteps further helped to improve F1 score by 11%. We also tried modeling the sequential features using transformer architectures (Vaswani et al., 2017) and bidirectional LSTM, but that did not give as much improvement as GRU. We believe this is because GRU uses fewer parameters and so it does not overfit the data. Adding the initial hidden

Model increment	Notation
Static features	A
Bootstrapping	B
Sequential features	C
With initial state h_0	D
Embedding for diagnosis	E
Multitasking	F

Table 6. *Short notation for increments*

Increments	AUC	Accuracy	Sensitivity	Specificity	Precision	F1	F1 Improvement
A	0.88	0.85	0.92	0.48	0.44	0.60	NA
A+B	0.9	0.87	0.94	0.56	0.55	0.69	17%
A+B+C	0.91	0.89	0.97	0.68	0.64	0.77	11%
A+B+C+D	0.95	0.92	0.97	0.71	0.72	0.83	7%
A+B+C+D+E	0.96	0.94	0.98	0.78	0.82	0.89	8%
A+B+C+D+E+F	0.97	0.96	1	0.88	0.88	0.94	5%

Table 7. *Performance improvement with successive inclusion of model increments*

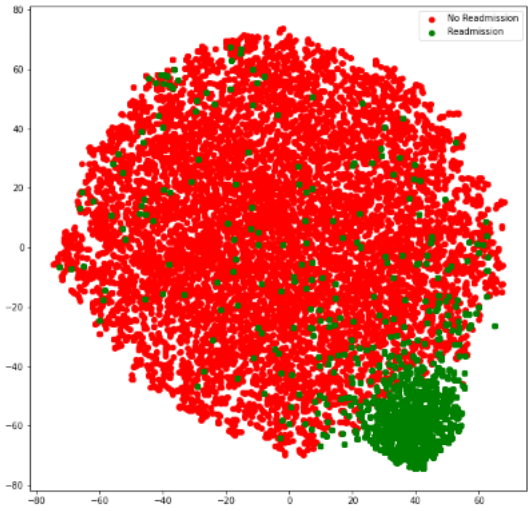

Figure 4. *t-SNE projection of embedding weights.*

state (A+B+C+D) of the GRU h_0 to a linear transformation of the user embedding resulted in a 7% improvement in F1 score. There was a increase in improvement in F1 score by 8% when we used diagnosis embedding on top of one-hot encoding (A+B+C+D+E). Figure 4 shows the projection of embedding weights in 2D space using t-SNE. Visual inspection of the weights shows that the embedding vectors of diseases with readmission tend to club together, and those that do not require readmission are separate proving that there is an implicit relation between diseases which cause readmission and which do not. Finally, adding multitask layer (A+B+C+D+E+F) to

the model improved the F1 score by 5% making it 0.94.

7 Conclusion

In this work, we have proposed a novel architecture called **S**equential **R**eadmission **P**redictor with **M**ultitasking (SRPM) that incorporates prior knowledge about the patients' information to classify whether a patient will have readmission after discharge. We achieved a significant improvement in the classification accuracy by bootstrapping the dataset to offset the imbalance towards the target. Inclusion of sequential features further enhanced the performance which is in line with previous works. Further, consideration of the patient embedding as an initial state to the GRU performed better than random initialization or zero initialization. This validates our reasoning to integrate the past information seamlessly into our model without overfitting. Inclusion of the embedding of the disease feature and one-hot encoding also improved the accuracy further. The t-SNE projection shows a clear grouping of diseases and goes in line with our intention to capture implicit connections between the diseases. Finally, using multitask learning with the auxiliary task of predicting the readmission days proves beneficial as we get much better performance than having a single task. We observed a gradual improvement with the addition of incremental changes proposed in our model. Some limitations of our study are that our model may not perform as well for new records which do not have a medical history. Since we replace unseen sequence tokens (which do not appear in the train set) that appear in the test set with out-of-vocabulary tokens, the models' performance is limited to that extent. We believe this recipe could also work for other tasks like mortality prediction, next disease prediction, etc., and it directs to an interesting future work.

References

Amarasingham, R., R. E. Patzer, M. Huesch, N. Q. Nguyen, and B. Xie (2014). "Implementing electronic health care predictive analytics: considerations and challenges." *Health Affairs* 33 (7), 1148–1154.

Balan U, M., M. Gandhi, S. Rammohan, et al. (2021). "Predicting Unplanned Hospital Readmissions using Patient Level Data." In: *Proceedings of the 54th Hawaii International Conference on System Sciences*, p. 3436.

Cheng, Y., F. Wang, P. Zhang, and J. Hu (2016). "Risk prediction with electronic health records: A deep learning approach." In: *Proceedings of the 2016 SIAM International Conference on Data Mining*. SIAM, pp. 432–440.

Deschepper, M., K. Eeckloo, D. Vogelaers, and W. Waegeman (2019). "A hospital wide predictive model for unplanned readmission using hierarchical ICD data." *Computer methods and programs in biomedicine* 173, 177–183.

Hahn, T., A. A. Nierenberg, and S. Whitfield-Gabrieli (2017). "Predictive analytics in mental health: applications, guidelines, challenges and perspectives." *Molecular psychiatry* 22 (1), 37–43.

Hillestad, R., J. Bigelow, A. Bower, F. Girosi, R. Meili, R. Scoville, and R. Taylor (2005). "Can electronic medical record systems transform health care? Potential health benefits, savings, and costs." *Health affairs* 24 (5), 1103–1117.

Jensen, P. B., L. J. Jensen, and S. Brunak (2012). "Mining electronic health records: towards better research applications and clinical care." *Nature Reviews Genetics* 13 (6), 395–405.

Khoury, M. J., M. Engelgau, D. A. Chambers, and G. A. Mensah (2018). "Beyond public health genomics: can big data and predictive analytics deliver precision public health?" *Public health genomics* 21 (5-6), 244–250.

Manga, J. and J. Sun (2020). "Patient Empowerment and Electronic Health Record Systems Sharing: A Design Perspective."

Manias, E., K. Gray, N. Wickramasinghe, and M. Manojlovich (2018). "Using electronic medical records to create big data and to communicate with patients—Is there room for both?" *Collegian* 25 (4), 467–469.

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

Nguyen, P., T. Tran, N. Wickramasinghe, and S. Venkatesh (2017). "`Deepr`: A Convolutional Net for Medical Records." *IEEE Journal of Biomedical and Health Informatics* 21 (1), 22–30. DOI: `10.1109/JBHI.2016.2633963`.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). "Scikit-learn: Machine Learning in Python." *Journal of Machine Learning Research* 12, 2825–2830.

Rajkomar, A., E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. (2018). "Scalable and accurate deep learning with electronic health records." *NPJ Digital Medicine* 1 (1), 1–10.

Si, Y. and K. Roberts (2019). "Deep patient representation of clinical notes via multi-task learning for mortality prediction." *AMIA Summits on Translational Science Proceedings* 2019, 779.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin (2017). "Attention is all you need." *arXiv preprint arXiv:1706.03762*.

Wager, K. A., F. W. Lee, and J. P. Glaser (2021). *Health care information systems: a practical approach for health care management*. John Wiley & Sons.

Wickramasinghe, N., H. Moghimi, and J. L. Schaffer (2017). "Leverage Healthcare Data Assets with Predictive Analytics: The Example of an Australian Private Hospital." In: *Decision Management: Concepts, Methodologies, Tools, and Applications*. IGI Global, pp. 823–837.

Wong, C. W., C. Chen, L. A. Rossi, M. Abila, J. Munu, R. Nakamura, and Z. Eftekhari (2021). "Explainable Tree-Based Predictions for Unplanned 30-Day Readmission of Patients With Cancer Using Clinical Embeddings." *JCO Clinical Cancer Informatics* 5, 155–167.

Yu, R., Y. Zheng, R. Zhang, Y. Jiang, and C. C. Poon (2019). "Using a multi-task recurrent neural network with attention mechanisms to predict hospital mortality of patients." *IEEE journal of biomedical and health informatics* 24 (2), 486–492.