Generating Electronic Health Records using Recurrent Generative Adversarial Networks

Aayush Saxena
Department of Computer Science
North Carolina State University
Raleigh, NC 27606
asaxena6@ncsu.edu

Abstract

The project aims to generate realistic Electronic Health Records using original Electronic Health Records (EHR) for both patients who suffer from Septic Shock and for patients who do not. The simulated EHRs should be indistinguishable from actual EHR's and should capture the true characteristics of the probability distribution of each feature for each class with some variance. The approach to achieve this task uses a Recurrent Generative Adversarial Network (RGAN) which is composed of a Long Short-Term Memory network (LSTM).

1 Introduction

Sepsis is a life-threatening organ dysfunction [Singer et al., 2016] and a leading cause of death in the United States. Sepsis, infection plus systemic manifestations of infection, is the leading cause of in-hospital mortality. The mortality rate can be reduced if there is a better system to take care of this.

Electronic Health Records (EHRs) are a large-scale and systematic collection of temporal health information of patients. It consists of various parameters across multiple events for each patient during each of their visits to the hospital. The EHRs can be used to model the progression of Sepsis using the temporal features for the patient across visits. However, getting access to Electronic Health Records(EHRs) is expensive and often a bottleneck in development. Moreover, due to the highly sensitive nature of medical data, its access is typically highly controlled.

Thus the motivation of the project is to use Electronic Health Records (EHR) to simulate realistic EHR's which are indistinguishable from actual EHR's. The newly generated data can be generated in abundance which will be useful for modeling the progression of the disease.Moreover, the generated data can be used to simulate a medical plan of action and test the efficacy of it, this can then be used as a proof of concept to apply on real cases.

2 Background

2.1 Generative Adversarial Network(GAN)

GAN's are neural networks which consist of two components called the generator and discriminator. Both the networks are trained in tandem alternatively. The task of the generator is to take noise as input and generate realistic data which is then fed into a discriminator. The task of the discriminator is to identify which data is real or fake, for which it takes a batch of generated samples and a batch of real samples. The generator loss is low if generator can generate data which can't be classified correctly by the discriminator. The discriminator loss is low if it is able to correctly distinguish between the real data and the generated data.

2.2 Wasserstein Generative Adversarial Network(WGAN)

GANs suffer from a vanishing gradient problem if the initial guess of the probability distribution of the generator is way off from the original. This happens since the GAN tries to minimize KL-Divergence, whose gradient will be near zero if the probability distributions are far from each other. WGANs try to solve this problem by using Earth Movers distance between the probability distributions as a cost function. The model tries to minimizes this Earth movers distance, but it requires the function to obey the Lipschitz constraint. Weight clipping is used to enforce the Lipschitz constraint.

2.3 Recurrent Generative Adversarial Network

Recurrent Generative adversarial networks[1] uses a similar framework as GANs but it replaces the generator and the discriminator by recurrent neural networks. So, the recurrent GAN can generate multivariate temporal data.

The discriminator for RGAN calculates loss using average negative cross entropy between the actual ground truth and the predictions for each time-step. Let the output of the discriminator be a vector $RNN_D(X) \in \mathbb{R}^d$ of length as the number of output features generated for each time-step. Let y_n be a vector of ones for real sequences and a vector of zeros for the synthetic sequences. The discriminator loss is equal to $D_{loss}(X_n, y_n) = -CE(RNN_D(X_n), y_n)$.

The generator aims to create samples from noise(Z) which can trick the discriminator in classifying them as real. Let $RNN_G(Z)$ be the output produced by the generator, then the loss will be equal to $D_{loss}(RNN_G(Z_n), 1)$

Long Short-Term Memory(LSTM) networks have been used as a RNN to train the model.

The architecture of the generator and the discriminator are shown in Figure 1 and 2 respectively.

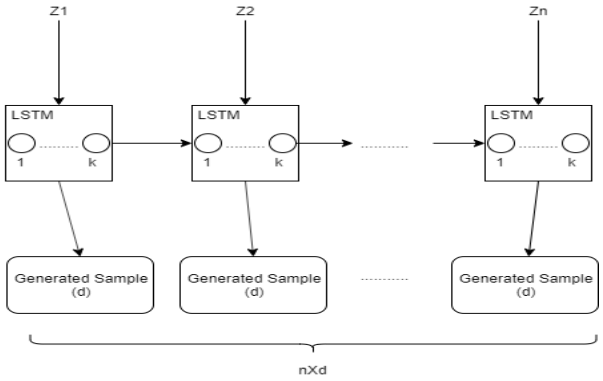


Figure 1: Generator Architecture

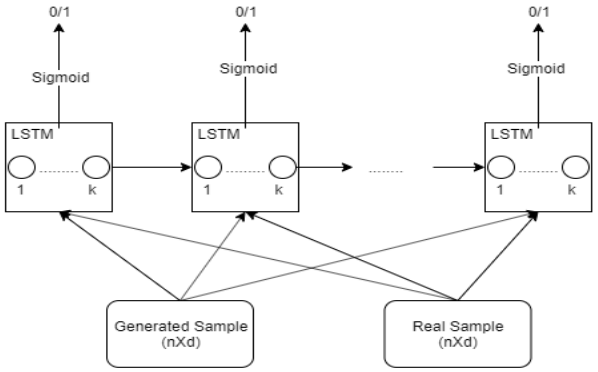


Figure 2: Discriminator Architecture

2.4 Recurrent Wasserstein Generative Adversarial Network(RWGAN)

This is similar to RGAN, but here we use Wasserstein distance as a cost function. Wasserstein or Earth movers distance is a distance metric between two probability distributions calculated as the minimum cost of transporting mass in converting one distribution to another. To implement RWGAN, weight clipping was required to enforce the Lipschitz constraint.

3 Methodology

3.1 Data Description

Two sets of data is used in the form of Electronic Health Records of patients who suffered Septic Shock and patients who didn't suffer Septic shock.The EHR data is a time series where each time-step for a patient has readings of vital features associated with it. The data consists of a Patient ID representing a unique patient and a Visit ID representing a unique visit. Within each visit the patient has multiple readings of various parameters at irregular time intervals called events. The features used for each event are HeartRate, RespiratoryRate, PulseOx, SystolicBP and Temperature which are all continous features.

3.2 Data Imputation

EHR data suffers from a lot of missingness in readings of its parameters. This is due to the fact that not all readings are taken at each event. To fill in the missing data, the data was grouped by both the patient Identifier and the Visit Identifier. Within each group each parameter was filled in using a forward fill mechanism. Once the forward fill is complete, the data is filled using a backward fill so that the remaining missing values are also filled in. The data from a previous visit for a patient is not used to forward fill the data in a subsequent visit. This is because patients visits occur at irregular time intervals and thus the readings at a previous visit may not be even close to the readings of the current visit.

3.3 Remove Duplicates

To avoid the violation of Independence assumption of the data, the data points which were duplicates within a Patient and Visit group were removed. The duplicates were removed based on all the features in the group which results in a data-set where for each unique visit we will have multiple unique readings of the various parameters.

3.4 Standardize sequence lengths for time-series

After some analysis of each of the time-series for the patient, the sequence length was chosen. This was based upon the frequency of data-points with sequence lengths greater than a number x. The final sequence length chosen was x = 30. Each time-series data point with less than x rows was removed and for each time-series data with more than x rows, only the last x rows were considered.

3.5 Normalization

The data was normalized using the mean and standard deviation of all the data in the time-series for each patient.

3.6 Training the RGAN

The RGAN was trained with the parameters mentioned in Table 1.

3.7 Training the RWGAN

A Recurrent Wasserstein Generative Adversarial network(RWGAN) was trained with the parameters mentioned in Table 2.

Table 1: RGAN Hyper-parameters

sequence length	30
Input Noise Dimension	10
Discriminator Training rounds	1
Generator Training rounds	10
Batch Size	56
Learning rate	0.001
hidden units in generator	100
hidden units in discriminator	100
epochs	120

Table 2: WRGAN Hyperparameters

lower clip bound	-0.01
upper clip bound	0.01
Discriminator Training rounds	5
Generator Training rounds	1
Learning Rate	0.00005
hidden units in generator	100
hidden units in discriminator	100
Batch Size	56
epochs	120
sequence length	30
Input Noise Dimension	10

4 Evaluation

4.1 Maximum Mean Discrepancy

GANs are effective since they train the generative network by directly comparing the generated distribution to the true one and hence, they implicitly learn the distribution of actual data. MMD checks if two samples of data come from the same distribution. The time series data is taken as a vector for comparison and the radial basis function (RBF) kernel using the squared Frobenius norm has been used between vectors.

$$K(x, y) = exp(-|x - y|^2/(2\sigma^2))$$

To select σ we maximize the t-statistic of the power of the MMD test between two distributions (Sutherland et al., 2016). $t = MMD^2/\sqrt{V}$, where V is the asymptotic variance of the estimator of MMD2.

4.2 Train on Synthetic, Test on Real(TSTR)

Once the GAN is trained for both Sepsis and non-Sepsis data-sets, we generate samples from the trained model for both of the groups. We train a classifier on the synthetic samples generated to classify the data as Sepsis or Non-Sepsis. The model trained here uses LSTMs with Binary Cross Entropy Loss and an Adam Optimizer. Once the model is trained on the synthetic data, we test this classifier on the original data sequences. If the model works well on the original sequences, that should indicate that the simulated data can be used instead of the real data for real applications.

4.3 Train on Real, Test on Synthetic(TRTS)

This approach is similar to TSTR, but here we train the model on the real data and then test it on the synthetic data. A similar classifier is trained here to classify sequences in Sepsis or Non-Sepsis category.

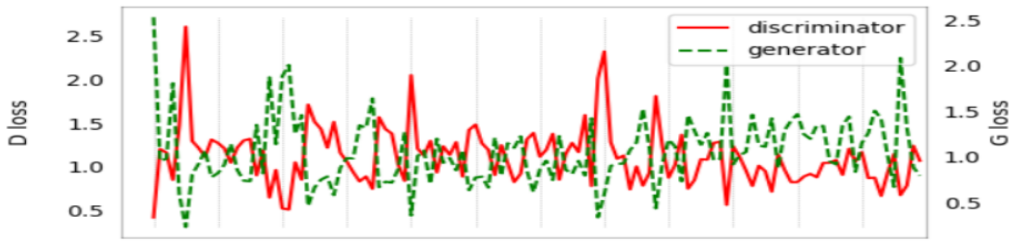
5 Results

5.1 RGAN

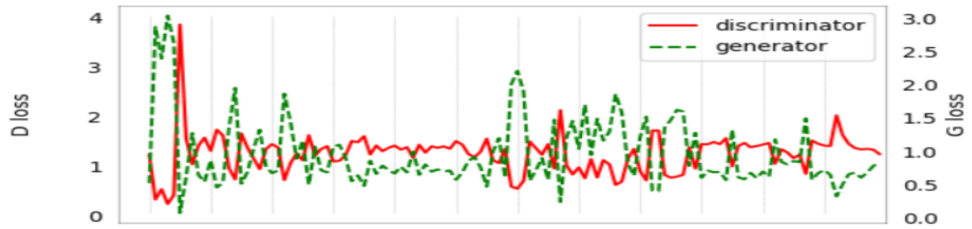
The following subsections show results of the trained RGAN model.

5.1.1 Model Loss - RGAN

The model was trained separately on Sepsis and non-Sepsis patients. The generator and discriminator loss of the model for Sepsis patients is shown in Figure 3a.The generator and discriminator loss of the model for Non-Sepsis patients is shown in Figure 3b.



(a) Sepsis

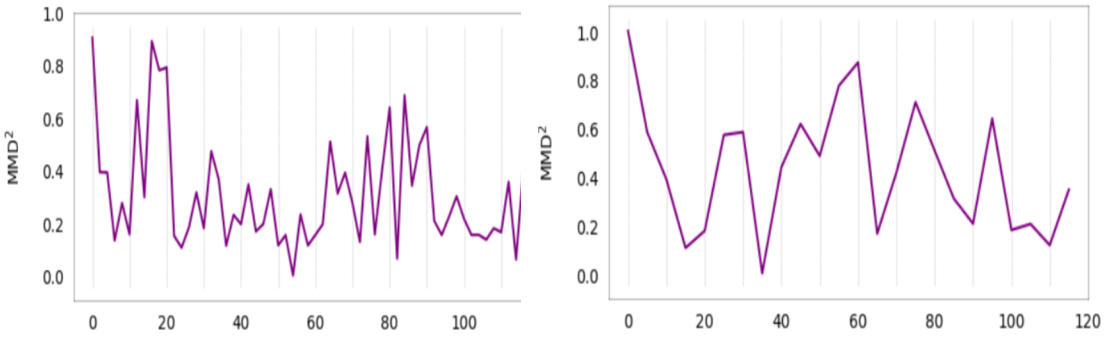


(b) Non-Sepsis

Figure 3: RGAN Loss per Epoch

5.1.2 Maximum Mean Discrepancy - RGAN

The evaluation metric MMD^2 for Sepsis and Non-Sepsis patients per epoch is shown in figure 4a and 4b respectively.



(a) Sepsis

(b) Non Sepsis

Figure 4: RGAN MMD^2 per Epoch

5.1.3 Train on Synthetic, Test on Real - RGAN

A LSTM classifier was used with the parameters mentioned in Table 3.

Table 3: RGAN - TSTR Classifier Parameters

LSTM cells	100
Activation	sigmoid
Discriminator Training rounds	1
Loss	Binary Cross Entropy
Optimizer	Accuracy
epochs	5

The results for TSTR are mentioned in Table 4

Table 4: RGAN - TSTR Classifier Results

Training Accuracy	100
Validation Accuracy	100
Testing Accuracy	57.6

Table 5: RGAN - TRTS Classifier Results

Training Accuracy	85.09
Validation Accuracy	80.23
Testing Accuracy	84.70

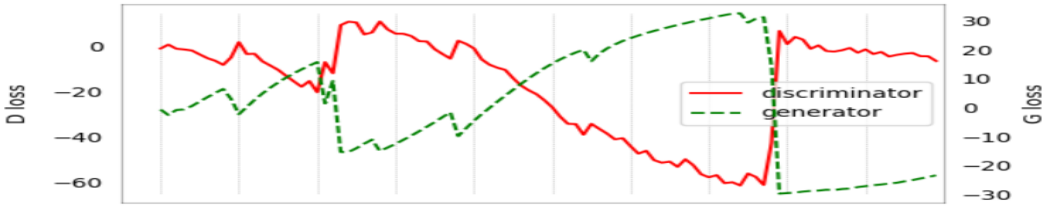
5.1.4 Train on Real, Test on Synthetic - RGAN

A LSTM classifier was used with parameters in Table 3 and the results for TRTS are in Table 5

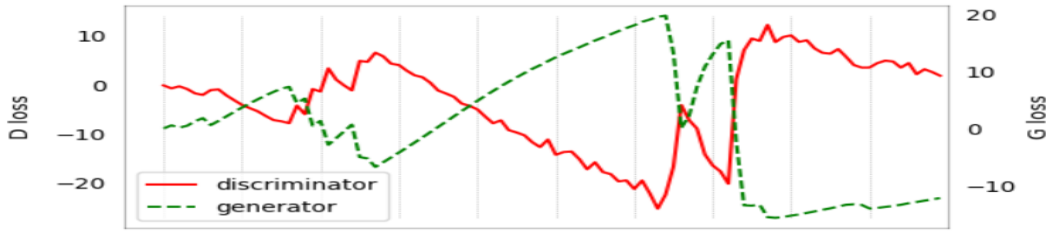
5.2 RWGAN

5.2.1 Model Loss - RWGAN

The generator and discriminator loss of the model for Sepsis patients is shown in Figure 5a and for Non-Sepsis patients is shown in Figure 5b.



(a) Sepsis

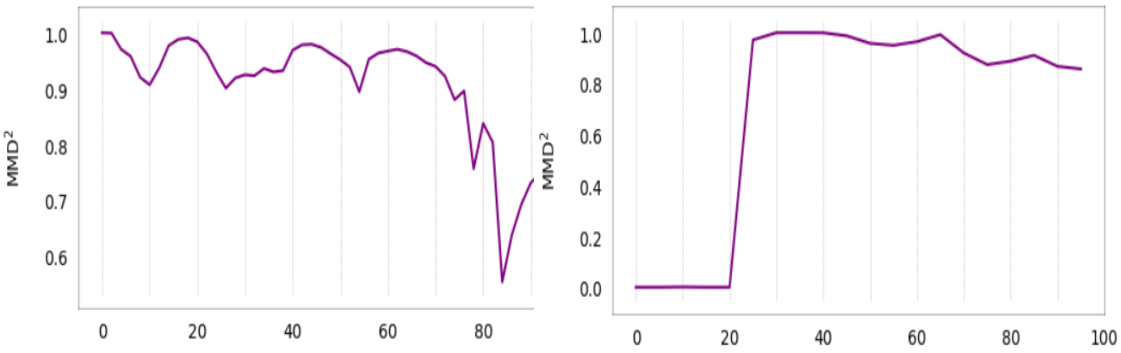


(b) Non-Sepsis

Figure 5: RWGAN Loss per Epoch

5.2.2 Maximum Mean Discrepancy - RWGAN

The evaluation metric MMD^2 for Sepsis and Non-Sepsis patients per epoch is shown in figure 6a and 6b respectively.



(a) Sepsis (b) Non Sepsis

Figure 6: RWGAN MMD^2 per Epoch

5.2.3 Train on Synthetic, Test on Real - RWGAN

A LSTM classifier was used with the parameters mentioned in Table 3.

The results for TSTR are mentioned in Table 6

Table 6: TSTR Classifier Results

Training Accuracy	100
Validation Accuracy	100
Testing Accuracy	52.78

5.2.4 Train on Real, Test on Synthetic - RGAN

A LSTM classifier was used with the parameters mentioned in Table 3.

The results for TRTS are mentioned in Table 7

Table 7: TRTS Classifier Results

Training Accuracy	83.53
Validation Accuracy	79.47
Testing Accuracy	83.05

6 Conclusion

In this project, a Recurrent Generative Adversarial network(RGAN) is trained for Sepsis and non-Sepsis patient data to generate realistic EHRs. Different evaluation metrics for GANs are explored to assess its performance. A different variant of RGAN is also trained using a Wasserstein cost function(RWGAN). A classifier is also trained using LSTMs to classify a timeseries as Sepsis or Non-Sepsis, which has been used for evaluation of the GAN. The results show oscillation in the generator/discriminator loss and hence indicate a lot of scope of improvement in terms of extensive hyper-parameter tuning to get better performance on the evaluation metrics and to get the model to

converge. As a future scope, a time-aware LSTM model can be used as the RNN component in the model to cater to the irregular time intervals in the data.

References

[1] Cristóbal Esteban, Stephanie L. Hyland, and Gunnar Rätsch. Real-valued (medical) time series generation with recurrent conditional gans, 2017.

[2] X. Yang, Y. Zhang, and M. Chi. Time-aware subgroup matrix decomposition: Imputing missing data using forecasting events. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 1524–1533, Dec 2018.

[3] Yuan Zhang, Xi Yang, Julie Ivy, and Min Chi. Attain: Attention-based time-aware lstm networks for disease progression modeling. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4369–4375. International Joint Conferences on Artificial Intelligence Organization, 7 2019.