

# CS598 DL4H Project Draft Spring 2023

Sagar Dalwadi and Murtaza Lodgher

{sagardd2, lodgher2}@illinois.edu

Group ID: 83

Paper ID: 12

Presentation link: TODO

Code link: TODO

## 1. Introduction

In the medical field, heart failure (HF) is a difficult disease to detect early on and diagnose to a patient as many of the early signs and symptoms are common across different kinds of common diseases. As a result, the “onset of HF is associated with a high level of disability, health care costs, and mortality” (Choi et al., 2017, pg. 362). A paper by Edward Choi, Andy Schuetz, Walter F Stewart and Jimeng Sun, published in the *Journal of the American Medical Informatics Association* explores how Recurrent neural network (RNN) models can be used to detect HF as early as possible, even more so than other deep learning methods, as they are tuned to temporal relations. Our team will attempt to reproduce the results from this paper and add extra parameters to see how the model will react to different circumstances.

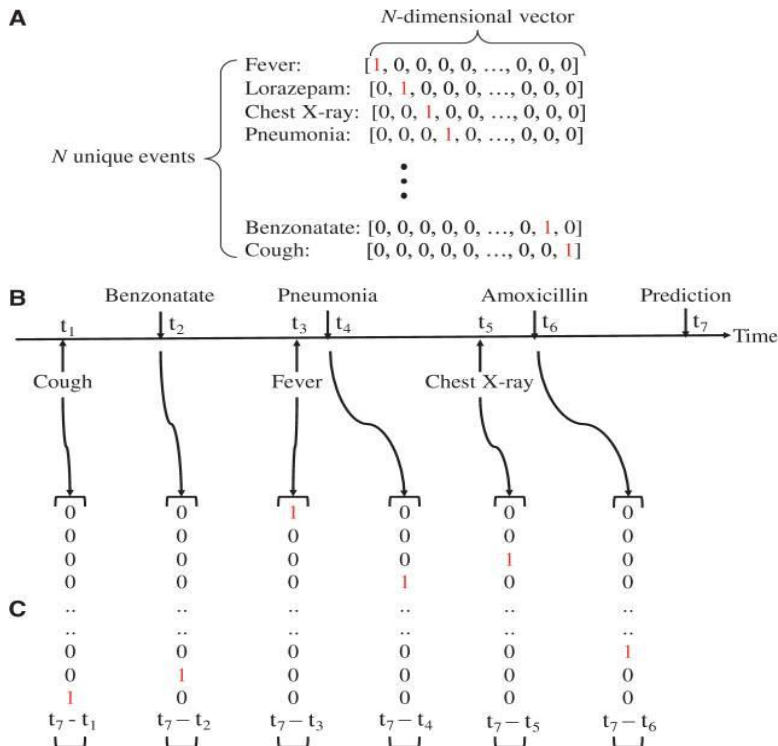
## 2. Scope of Reproducibility

The claims from this paper we have chosen to reproduce are as follows:

- RNN models using gated recurrent units (GRUs) were able to yield a higher area-under-the-curve (AUC) for predicting HF diagnoses through electronic health record (EHR) data over conventional deep learning methods, like logistic regression, SVM and KNN.

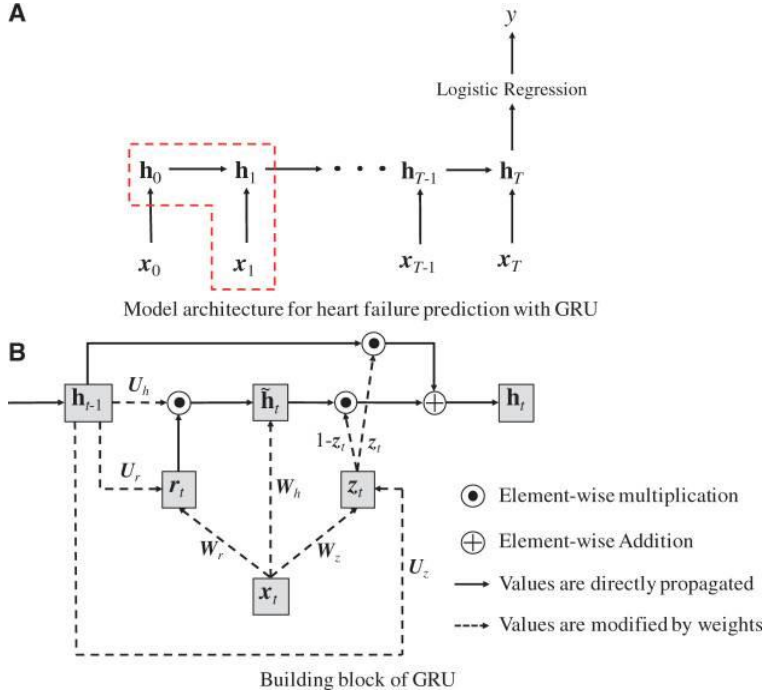
## 3. Methodology

### 3.1. Model Descriptions



To construct the GRU model, we first take the sequence of symptoms and convert them into hot-vectors using labels ranging from 0 - 1 to alter the dimension. We then arrange the vectors in order based on the time each symptom occurred, before a prediction was made. This concept is demonstrated in the image taken directly from the paper, where section A is converting the symptoms to hot-vectors, section B arranges the vectors based on time, assuming a prediction is made at time  $t_7$ , and section C appends the time feature at the end of each hot vector:

This vector is then used as the input vector  $x_t$  for the GRU, where  $t$  is timestep for the number of clinical visits  $T$ . The vector is transformed and stored into a hidden layer  $h$ , whose state changes over time as more input vectors are added to  $h$  in each timestep. After the final input vector  $x_T$  has been added to  $h$ , a logistic regression function is applied and it produces the scalar vector  $y$ , representing the estimated patient-specific score for future diagnosis of HF. The model structure is illustrated in section A of the image below:



As shown in Section B of the image, The GRU model has four components:  $z_t$ : the update gate at time step  $t$ ,  $r_t$ : the reset gate at time step  $t$ ,  $\tilde{h}_t$ : the intermediate memory unit at time step  $t$  and  $h_t$ : the hidden layer at time step  $t$ .

Mathematical formulation of GRU components:

$$\begin{aligned} z_t &= \sigma(W_z x_t + U_z h_{t-1} + b_z) \\ r_t &= \sigma(W_r x_t + U_r h_{t-1} + b_r) \\ \tilde{h}_t &= \tanh(W_h x_t + r_t \odot U_h h_{t-1} + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \end{aligned}$$

The logistic regression model was applied to the final state of the hidden layer as formulated below:

$$y = \sigma(w^T h_T + b)$$

And the loss function used for minimization is:

$$\text{Loss} = -\sum_{i=1}^P (c^{(i)} \log y^{(i)} + (1 - c^{(i)}) \log (1 - y^{(i)}))$$

### 3.2. Data Descriptions

Currently, we are running our implementation of the code based on the synthetic data given in the code repository attached in the paper (link: [https://github.com/mp2893/rnn\\_predict.git](https://github.com/mp2893/rnn_predict.git)). This data is split into 4 sections: Sequence data (sequence of symptoms provided as medical codes), Times data (data at what time these symptoms occurred), Label data (data for changing the dimensionality of each symptom at a given time), and Embedded data (pre-trained medical code data). Our eventual goal is to use real data sourced from actual medical institutions (from physionet, for example) similarly structured to the synthetic data in the codebase.

### 3.3. Hyperparameters

The following hyperparameter settings were used to train GRU model:

- L2 regularization: 0.001
- Hidden layer size: 100
- Max epoch: 30 (For the initial run we kept the max epoch low to have a successful run)

These hyperparameters were referenced from the supplementary data provided in the paper.

### **3.4. Implementation**

As an initial step towards reproducing the results for the proposed model from the paper and to further implement our proposed ablations,

- The first step of our current implementation involves using the original code base provided in the paper and making the necessary updates to the code to make it compatible with the latest version of Python including necessary updates to the libraries used such that we are able to successfully run the model against the data and produce the results.
- We utilized the synthetic data provided and passed it through the model for testing purposes and successfully generated the AUC for each epoch.
- Our thought process behind following this implementation approach was to first understand how the GRU model is constructed and how it performs against the synthetic data.
- Now that we have successfully tested the implementation of the GRU model, the next step in our implementation would be to utilize the EHR data from PhysioNet and transform it in the form required by the model as described earlier. And test the processed data with the existing implementation of GRU model as well as implement the proposed ablations and re-test the model with the processed data.

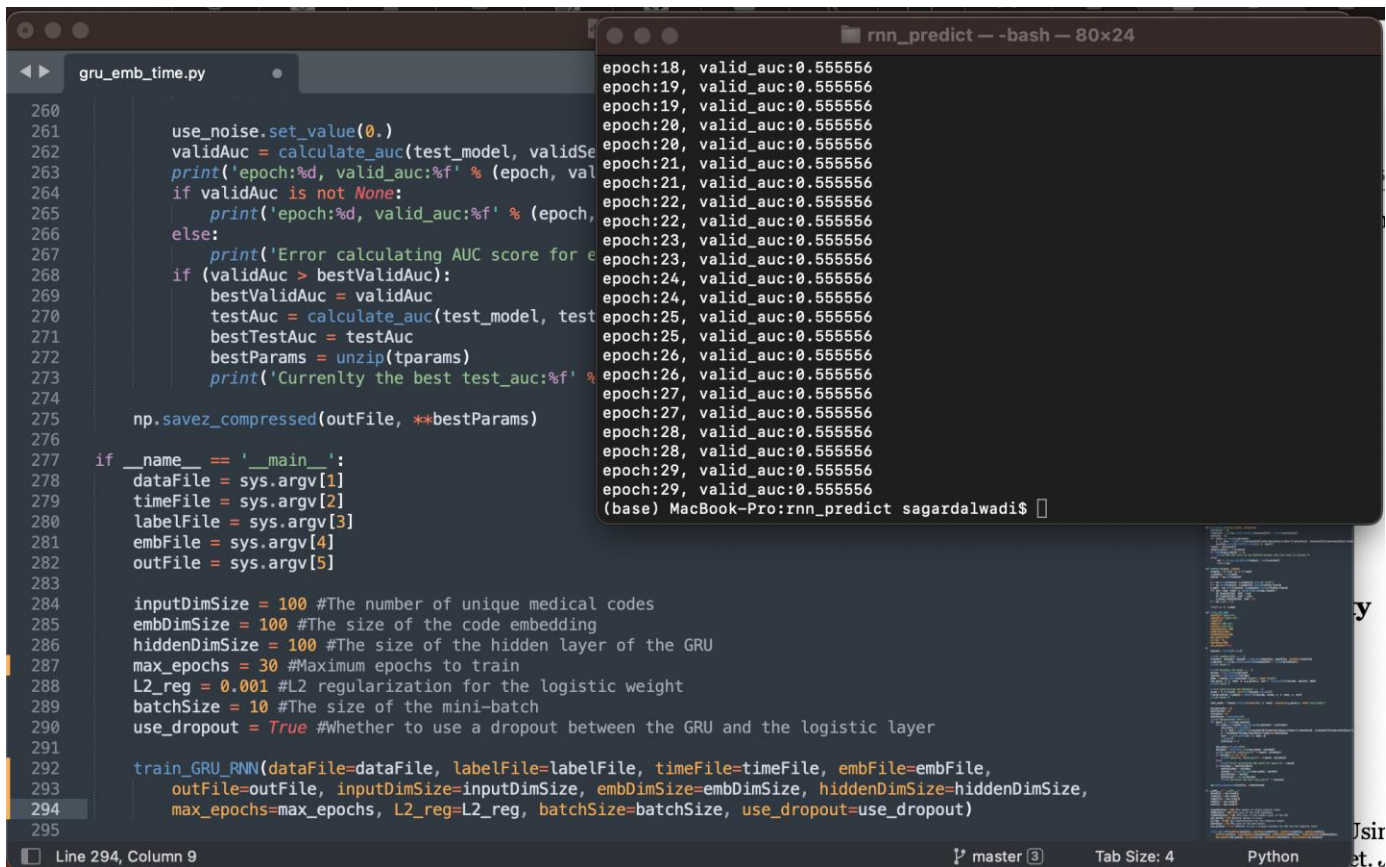
### **3.5. Computational Requirements**

At this initial stage of implementation, we do not have any additional computation requirements and we were able to utilize the personal machine with good enough specs. However, as we progress through this with additional ablations and trying to improve the performance, we may utilize additional computation capacity from third party providers like AWS or others to support the execution of our model

## **4. Results**

We have used synthetic test dataset provided by authors to test the implementation of GRU model.

As mentioned earlier our initial experiment was to run the base GRU model provided in paper with the synthetic test data and with that we have obtained the AUC of 0.555556 as shown below:



```
gru_emb_time.py
260
261 use_noise.set_value(0.)
262 validAuc = calculate_auc(test_model, validSe
263 print('epoch:%d, valid_auc:%f' % (epoch, val
264 if validAuc is not None:
265     print('epoch:%d, valid_auc:%f' % (epoch,
266 else:
267     print('Error calculating AUC score for e
268 if (validAuc > bestValidAuc):
269     bestValidAuc = validAuc
270     testAuc = calculate_auc(test_model, test
271     bestTestAuc = testAuc
272     bestParams = unzip(tparams)
273     print('Currently the best test_auc:%f' %
274
275 np.savez_compressed(outFile, **bestParams)
276
277 if __name__ == '__main__':
278     dataFile = sys.argv[1]
279     timeFile = sys.argv[2]
280     labelFile = sys.argv[3]
281     embFile = sys.argv[4]
282     outFile = sys.argv[5]
283
284     inputDimSize = 100 #The number of unique medical codes
285     embDimSize = 100 #The size of the code embedding
286     hiddenDimSize = 100 #The size of the hidden layer of the GRU
287     max_epochs = 30 #Maximum epochs to train
288     L2_reg = 0.001 #L2 regularization for the logistic weight
289     batchSize = 10 #The size of the mini-batch
290     use_dropout = True #Whether to use a dropout between the GRU and the logistic layer
291
292     train_GRU_RNN(dataFile=dataFile, labelFile=labelFile, timeFile=timeFile, embFile=embFile,
293                   outFile=outFile, inputDimSize=inputDimSize, embDimSize=embDimSize, hiddenDimSize=hiddenDimSize,
294                   max_epochs=max_epochs, L2_reg=L2_reg, batchSize=batchSize, use_dropout=use_dropout)
295
rnn_predict --bash -- 80x24
epoch:18, valid_auc:0.555556
epoch:19, valid_auc:0.555556
epoch:19, valid_auc:0.555556
epoch:20, valid_auc:0.555556
epoch:20, valid_auc:0.555556
epoch:21, valid_auc:0.555556
epoch:21, valid_auc:0.555556
epoch:22, valid_auc:0.555556
epoch:22, valid_auc:0.555556
epoch:23, valid_auc:0.555556
epoch:23, valid_auc:0.555556
epoch:24, valid_auc:0.555556
epoch:24, valid_auc:0.555556
epoch:25, valid_auc:0.555556
epoch:25, valid_auc:0.555556
epoch:26, valid_auc:0.555556
epoch:26, valid_auc:0.555556
epoch:27, valid_auc:0.555556
epoch:27, valid_auc:0.555556
epoch:28, valid_auc:0.555556
epoch:28, valid_auc:0.555556
epoch:29, valid_auc:0.555556
epoch:29, valid_auc:0.555556
(base) MacBook-Pro:rnn_predict sagardalwadi$
```

First of all, we understand that an AUC of 0.555556 is not particularly high. In fact, an AUC of 0.5 means that the model is performing no better than random chance. So, while the model may be consistently producing a slightly better result than random chance, it's not performing well enough to be useful in detecting heart failure onset.

However, it's also important to keep the fact in mind that the data used for this experiment is only test data and is not a true representation of the broader population of required features from EHR data.

There are a few things we could try to improve our model's performance:

- Firstly, we will utilize the actual EHR datasets from the PhysioNet database as mentioned earlier to be able to evaluate the true performance of the model. We will construct the features based on the data gathered from these datasets, as this data may help the model to better capture the patterns and relationships in the data.
- Another possibility is that the features used to train the model may not be informative enough for detecting heart failure onset. It may be worth exploring other features that are more strongly associated with the development of heart failure.

## References

1. Choi, E., Schuetz, A., Stewart, W. F., & Sun, J. (2017). Using recurrent neural network models for early detection of heart failure onset. *Journal of the American Medical Informatics Association : JAMIA*, 24(2), 361–370. <https://doi.org/10.1093/jamia/ocw112>