**CS598 DL4H Project Report Spring 2023**Sagar Dalwadi and Murtaza Lodgher  
{sagardd2, lodgher2}@illinois.edu  
Group ID: 83  
Paper ID: 12

Presentation link: <https://youtu.be/N62x_KYRfrQ>

Code link: <https://github.com/malodgher/dlh-project-sp23.git>

1. **Introduction**

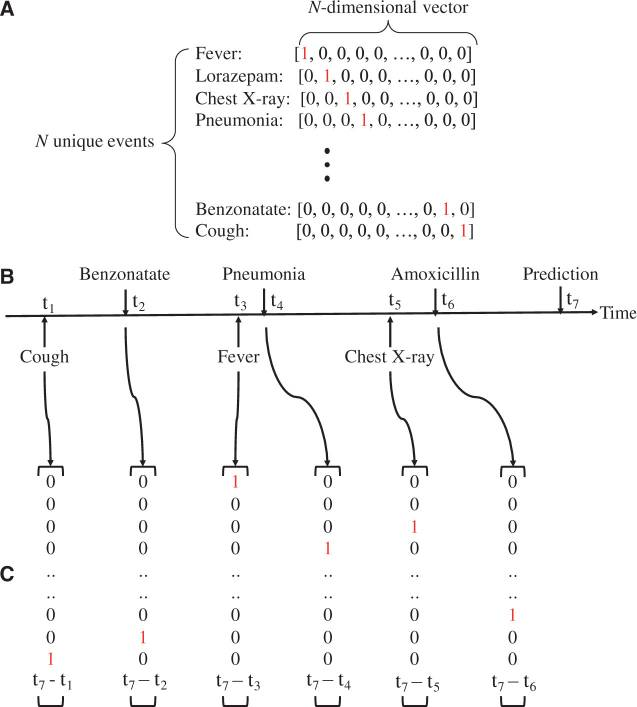
In the medical field, heart failure (HF) is a difficult disease to detect early on and diagnose to a patient as many of the early signs and symptoms are common across different kinds of common diseases. As a result, the “onset of HF is associated with a high level of disability, health care costs, and 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.

1. **Scope of Reproducibility**

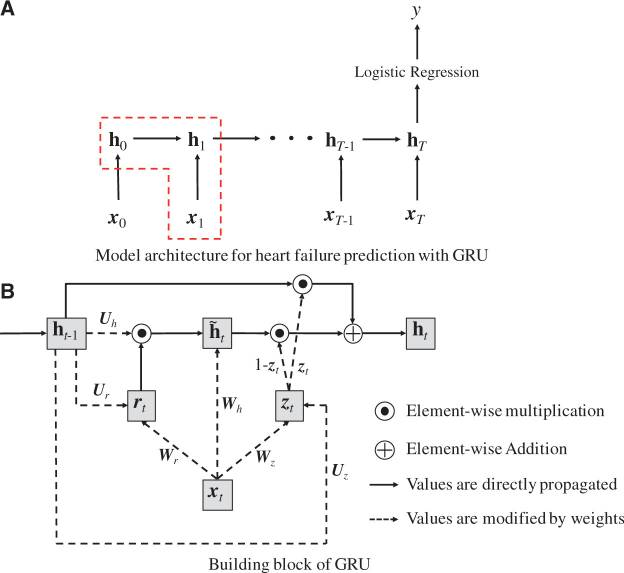
The claims from this paper we have chosen to reproduce are as follows:

* 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.

1. **Methodology**
   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 t7, and section C appends the time feature at the end of each hot vector:

This vector is then used as the input vector 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 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: : the update gate at time step 𝑡, : the reset gate at time step 𝑡, : the intermediate memory unit at time step 𝑡 and : the hidden layer at time step 𝑡.

Mathematical formulation of GRU components:

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

𝑦 =

And the loss function used for minimization is:

Loss = -

* 1. **Data Descriptions**

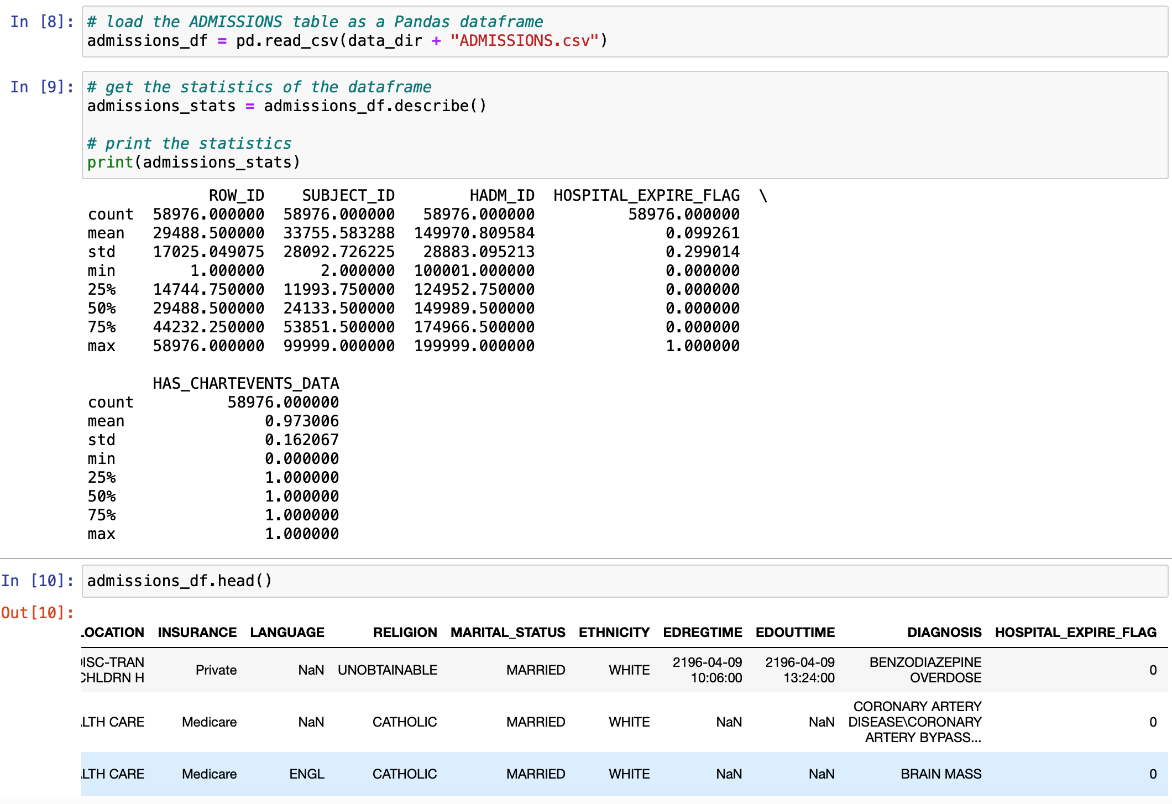
Initially we ran our implementation based on the synthetic data provided by authors and later we switched to the real word data as described below.

The datasets are extracted from PhysioNet website: (you need to have "CITI Data or Specimens Only Research" training completed & approved before you can access the data): <https://physionet.org/content/mimiciii/1.4/>

This Data includes following:

1. Admissions: gives information regarding a patient’s admission to the hospital
2. Patients: Provides information on each patient
3. Diagnoses ICD: Contains ICD diagnoses for patients
4. ICU Stays: defines a single ICU stay
5. Lab events: Contains all laboratory measurements for a given patient

More information on dataset, fields etc can be found at: <https://mimic.mit.edu/docs/iii/tables/>

The ICD diagnosis code data was essential in deriving the target variable of heart failure indicator. The Patients data, diagnoses data as well as the lab events for patients were critical in initializing the One hot vector, building and running data through the GRU for HF prediction. Here are some statistics of the data we used (more details can be seen [here](https://github.com/malodgher/dlh-project-sp23/tree/master/StudentCode/Input%20Data%20Stats)). The statistics of the hospital admissions data, the statistics of the patient data and the lab events data are shown:

Text

Description automatically generated

Text

Description automatically generatedChart, bar chart

Description automatically generated

After deriving HF indicator for patients, above bar chart was created where the resulting bar chart displays the frequency of heart failure cases for each gender, with each bar representing a different gender. This type of chart is useful for comparing the number of cases between different categories and identifying any differences or patterns between them.

We also created a set of histograms to visualize the distribution of the data for each variable in the input data frame. Charts can be seen [here](https://github.com/malodgher/dlh-project-sp23/tree/master/StudentCode/Input%20Data%20Stats) The resulting figure shows a set of subplots, with each subplot displaying a histogram for a different variable. This can help to provide insight into the distribution of the data, such as whether it is normally distributed or skewed, and identify any outliers or patterns in the data.

* 1. **Hyperparameters**

The following hyperparameter settings were used to train GRU model:

* Units: the number of units/neurons in the GRU layer. It is set to 32
* Activation: the activation function used in the GRU and Dense layers. The GRU layer uses the 'relu' activation function, while the Dense layers use the 'relu' and 'sigmoid' activation functions.
* Dropout: the dropout rate used in the Dropout layer. It is set to 0.2
* Optimizer: the optimization algorithm used for training the model. In this code, it is set to Adam optimizer with a learning rate of 0.001.
* Loss: the loss function used to compute the difference between the predicted and actual values. In this code, it is set to 'binary\_crossentropy'.
* Epochs: the number of times the entire training dataset is passed through the neural network during training. It is set to 3 for testing
  1. **Implementation**

As mentioned earlier in the draft, initially we utilized author’s code along with synthetic data to be able to gain understanding of the model and later we implemented our own code with real world data. The code we used for our implementation, while going by the framework of the original code base (link is [here](https://github.com/mp2893/rnn_predict.git)), is entirely written by members of our team and it implements the ablations laid out in our project proposal as well as below:

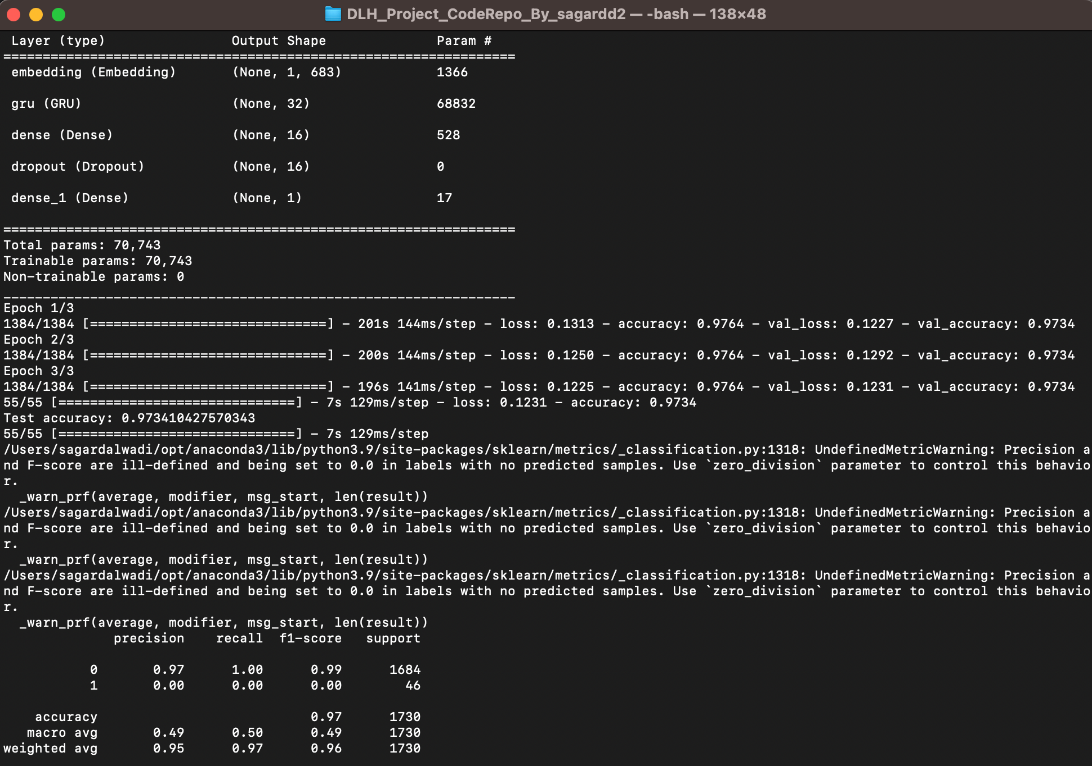
* The first step of our implementation involves reading in the data files, describing the statistics for each data file, and merging them into one pandas data frame.
* This data frame was then converted into One hot vector encoding.
* We then implemented the GRU model using the keras library’s Sequential class. Our ablation comes from the activation we used when initializing the GRU. For one GRU model, we set the activation function to ReLU and for another model, we set the activation function to Sigmoid. Our team wanted to see if different activation functions would lead to different results (drastic or otherwise) than the results given in the original paper.
* A model with ReLU: it defines a neural network model using Keras with three layers: an Embedding layer, a GRU layer, and two dense layers. The Embedding layer is used to map the input data to a lower-dimensional space, while the GRU layer is a type of recurrent neural network that can model sequential data. The two dense layers are fully connected layers that add nonlinearity to the model. The activation functions used in the model are ReLU for the GRU and the first dense layer, and Sigmoid for the final dense layer. A dropout layer is also included to help prevent overfitting
* Finally, we split the data in training & testing and ran through the GRU models and outputted the resulting accuracy.
  1. **Computational Requirements**

We used the personal machine with 2.3 Ghz 8-Core, 16 GB 2400 MHz DDR4, UHD Graphics 630 specifications to execute our implementation. Initially, there were some computational issues with our code, with timeouts and such, and we were considering using an outside vendor like AWS to run our code on. However, we were able to run our code by modifying some of the parameters:

* Code with Sigmoid activation, 3 epochs, 1000 input records: To train 1 epoch, takes ~3 hours
* Code with ReLu activation, 3 epochs, 15000 input records: To train 1 epoch, takes ~5 min
* Code with ReLu activation, 3 epochs, full data (~700k): To train 1 epoch, takes ~9 hours

1. **Results**

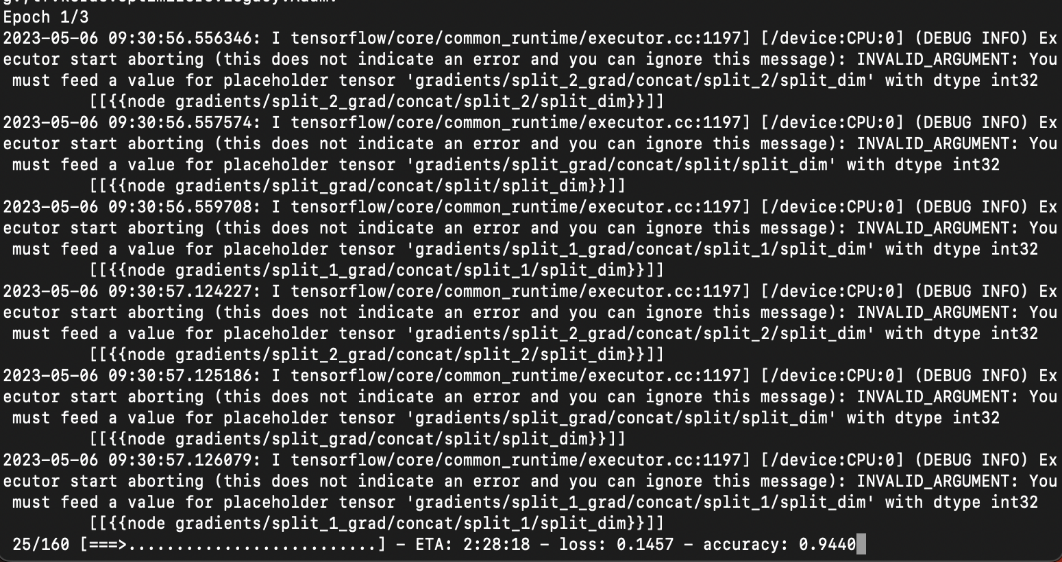
The results that the code was able to output, regardless of our ablations, was able to match the main claim in our scope of reproducibility, that RNN models are able to yield a larger AUC for predicting HF than other traditional models.



**Result 1**

The results from the image on the right are from a run of the data with the GRU initialized to a ReLU activation function. The accuracy of the model is outputted to be at 0.97, with a loss at about 0.12 – 0.13 per epoch. While the highest accuracy given from the paper with a one-hot vector utilization was about 0.75, this still represents a large factor of accuracy for using RNN for predictive analysis of HF. The difference in accuracy may be attributed to the difference in data used as well as input features available in data we used vs data authors used.

**Result 2**



The results from the image on the left are from a run of the data with the GRU initialized to a Sigmoid activation function. The accuracy of the model is outputted to be at 0.94, with a loss at about 0.14 – 0.15 per epoch. While a slightly lower accuracy than the ReLU activation function, this still supports the claim that, with any activation function, a RNN model will yield the highest accuracy than compared to conventional models.

1. **Discussion**
   1. **What was easy**

One of the easier aspects for our team when developing or implementing was that, the GRU model description was quite detailed, and we were able to develop our model very similarly to what was described in the paper. Another benefit that our team had was that we had a code base written by the authors that our team could reference for any technical code implementations.

* 1. **What was difficult**
* One of the difficult aspects of creating our implementation was trying to find data that had a similar structure to the synthetic data given in the author’s attached code repository. The actual data that the author’s mentioned in the paper was Sutter Palo Alto Medical Foundation, which may have required some amount of payment to access the same data, so we had to go with data from Physio.Net that is free data from various medical institutions and we had to make sure that we could derive the same data structure as the synthetic data from this free data.
* Understanding of the data is very critical. Without understanding what data you are dealing with, it is very difficult to build any models or even identify important features.
* Another issue our team ran into was that we couldn’t reproduce the number of epochs that were done in the original code due to limited hardware. As mentioned earlier, this problem was resolved by modifying the parameters in code somewhat, i.e., reducing the number of epochs generated when training.
  1. **Recommendations for reproducibility**
* Make sure that one has powerful hardware to run the same number of epochs as the original code
* If possible, try to collect the data from the original source than through free resources to get the most optimum results
* Know that one doesn’t need to use the exact same libraries as described in the original code. If there are libraries that can produce and train a model more efficiently, go ahead and use those.

1. **Communication with the original authors**

Our team did not have any communication with the original authors of the paper. Our implementation and findings were based solely on our team member’s own interpretations of the paper.

1. **Team members contribution statement**

* The team members were in constant communication with each other and were able to update each other on their work status. Both team members had some part to play in making sure the materials needed to be created and submitted were done so accordingly.
* The code implementation was written by Sagar Dalwadi. Initially, our decision was to have both members work on the different ablations (with Sagar working on the ReLU code and Murtaza working on the Sigmoid code), however, Murtaza was running into many issues with his implementation, with sections not properly connected and needed to be rewritten entirely, so we decided to tweak Sagar’s implementation to account for the Sigmoid ablation.
* Zoom meetings and the mobile group chat were facilitated by Murtaza Lodgher. Both the initial project draft and the final draft were written jointly. The presentation was also a joint effort between our two team members.

**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>
2. <https://physionet.org/content/mimiciii/1.4/>
3. <https://mimic.mit.edu/docs/iii/tables/>
4. <https://github.com/mp2893/rnn_predict>