Reproducibility Project Instructions for CS598 DL4H in Spring 2023

Joanna Regan
joanna.regan@illinois.edu
NetID: joannar2

Group ID: 206
Paper ID: 182
Presentation link: https://youtu.be/KOWy2eBMmfM
Code link: https://github.com/joanna-regan/CS598_DL4H_StageNet
Extra Credit NB link: https://github.com/joanna-regan/CS598_DL4H_StageNet/blob/main/ExtraCredit_Notebook.ipynb

1 Introduction

Throughout a patient's stay in the ICU, healthcare professionals are able to collect various observational and test results that contribute to that patient's EHR record. Existing deep learning techniques (Inci M Baytas and Zhou, 2017) have shown success in predicting patient health risk by assessing the longitudinal EHR data to learn disease patterns.

However, existing methods assume that the progression of the disease is indicated by a steady change in observed conditions over time, i.e. that the more time that passes, the greater change we will see in health status. The novel approach proposed by the current paper looks to remove the assumption that disease progression is steady over time, and rather looks to investigate whether a rapid change to health status occurs at some point in time as an indicator of disease progression and then leveraging patterns within those disease stages to inform prediction.

To do so, the original authors propose a Stage-aware Neural Network model they call StageNet (Gao et al., 2020). This model combines a stage-aware LSTM module with a stage-adaptive convolution module. The goal of StageNet is to identify the points in time where health status changes rapidly (i.e., enters a new stage), and to dynamically learn information about the patterns within each stage to help make predictions about a patient's health risk.

We focus StageNet on a binary health risk prediction task (specifically, decompensation) using MIMIC-III EHR dataset.

2 Scope of Reproducibility

The main goal of the original paper is to produce a model that can dynamically identify and leverage disease stage information to make better predictions on patients' health risk. Consequently, we aim to reproduce the full StageNet model as outlined by the original authors in their paper and their provided github code.

We also look to address their claims that the stage-aware components of the LSTM offer improvement over a vanilla LSTM, and that the stage-adaptive convolution module enhances the capabilities of the stage-aware LSTM alone. Consequently, we look to create two ablations that focus on these components to test whether they are salient for prediction.

Below we indicate the main claims we intend to test throughout the reproducibility project:

- StageNet trained with MIMIC-III EHR data will achieve 10% higher AUPRC and min(Re, P+) than baseline models on decompensation risk prediction task.

- Reduced models StageNet-I and StageNet-II will still achieve higher AUPRC, AUROC, and min(Re, P+) than all baseline models on decompensation risk prediction task.

3 Methodology

3.1 Model descriptions

In this section, we describe the overall architecture of the StageNet model as well as the individual components, including the stage-aware LSTM, the stage-adaptive convolution, and the prediciton module. We also describe the architecture for two ablation studies, StageNet-I and StageNet-II.

We show the overall architecture of StageNet in Figure 1, with the same caption as given in the original paper.

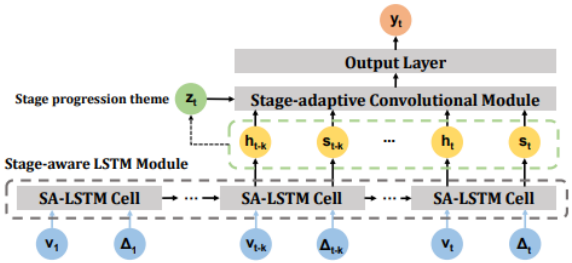

Figure 1: StageNet model: at the t-th timestep, the stageaware LSTM takes current visit v_t and elapsed time Δt as input to calculate current hidden state h_t and current stage variation s_t. Then the hidden states in the observation window K will be fed into the stage-adaptive convolutional module. The convolutional module will extract progression patterns at the current stage and re-calibrate these patterns using the progression theme z_t. Then the module will use recalibrated patterns to predict health risk $\hat{y}_t$.

3.1.1 Stage-Aware LSTM

Data is first fed step by step through a modified Long Short Term Model. The goal of this stage-aware LSTM variation is to differentiate historical state information by recent history versus old history. A change in state information that occurred recently indicates that the disease has progressed to a new stage.

To implement this, the authors introduce two new gates to the LSTM cell - a master forget gate representing old history and a master input gate representing recent history. These gates are two learned masking vectors which can be used to store information about whether the patient's health status has just entered a new stage, which we designate at s_t.

Rather than use an existing package like PyTorch's nn.LSTM(), we implement the LSTM from scratch so we can feed it one time step (or visit) at a time and use the incremental outputs as inputs to the convolution operation.

We use an input dimension of 93, corresponding to the 76 columns of the MIMIC-III database and 17 columns of the ESRD study (the latter of which is omitted from the current reproducibility study and filled with zeros during implementation).

We show the architecture of the stage-aware LSTM cell in Figure 2, with the same caption as given in the original paper.

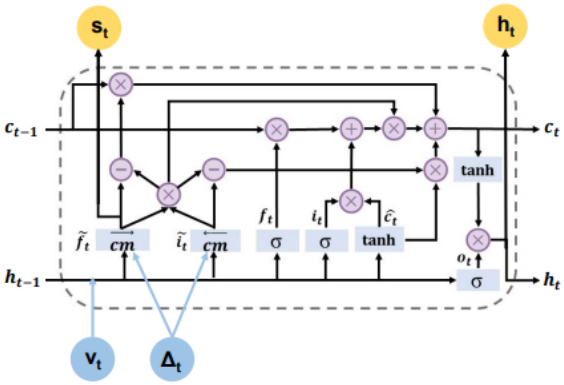

Figure 2: The structure of the stage-aware LSTM cell

3.1.2 Stage-Adaptive Convolution

After the stage-aware LSTM, the hidden states and stage variation captured within the observation window are fed into the stage-adaptive convolution module in order to learn the disease progression patterns.

Two sub-processes take place - learning stage progression themes at the current stage and recalibrating progression patterns.

To learn progression themes, the hidden states and stage variations are fed into a 1D convolution kernel to obtain u_t. To recalibrate the patterns, the hidden state and stage variation information is fed through a fully connected layer, followed by ReLu activation, another fully connected layer, and finally sigmoid function to produce importance vector x_t. Finally, we apply the recalibrated patterns to the output of the convolution by multiplying x_t and u_t.

We show the architecture of the stage-adaptive convolution module in Figure 3, with the same caption as given in the original paper.

3.1.3 Prediction

Finally, we take the output of the convolutional module as input to a Linear layer and output a binary label $\hat{y}_t$. We use cross-entropy loss and the Adam algorithm for optimization.

3.2 Ablations: StageNet-I and StageNet-II

In the first ablation, we replace the Stage-Aware LSTM cell with a vanilla LSTM implementation. We continue with the stage-adaptive convolution, but since we no longer have the f master gate we do not reweight the inputs to the convolution at each time step. Instead, we use the average of the output

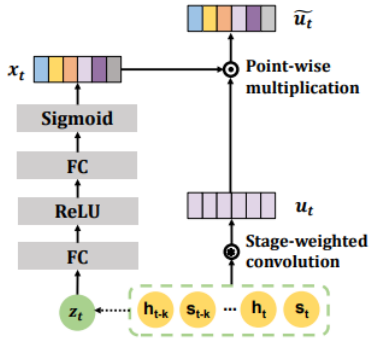

Figure 3: Stage-adaptive convolutional module: The module takes historical hidden state h_t and stage variation s_t within the observation window K as input, and learn progression patterns u_t using stage-weighted convolution operation. These patterns are re-calibrated to emphasise the most informative patterns at the current stage via extracting the current progression theme z_t.

h_t from the vanilla LSTM during the observation window as input to recalibrate the progression patterns.

In the second ablation, we only use the stage-aware LSTM and remove the convolution operation entirely. We use the output hidden state vector directly for prediction.

3.3 Data descriptions

The data used in this study includes the MIMIC-III EHR (Johnson, 2016) obtained from PhysioNet (Goldberger, 2000).

Data was downloaded from the PhysioNet MIMIC-III website [1] after obtaining permission for use. After downloading, data was extracted and preprocessed using modified instructions implemented by Harutyunyan et. al. (Harutyunyan et al., 2019), found at https://github.com/YerevaNN/mimic3-benchmarks/.

The final clinical time-series EHR data is composed of input sampled every two hours (referred to as a "visit") for the duration of a patient's stay in the ICU. During each visit (i.e. at each time step), we obtain information for 17 clinical variables, 5 of which are categorical (Capillary refill rate, Glasgow coma scale eye opening, Glasgow coma scale motor response, Glasgow coma scale verbal response, and Glasgow coma scale total) and the remainder of which are continuous (diastolic blood pressure, fraction of inspired oxygen, glucose, heart rate, height, mean blood pressure, oxygen saturation, respiratory rate, systolic blood pressure, temperature, weight, and pH). We discretize and standardize the continuous variables and apply one-hot encoding to the categorical variables. The final EHR data representation for a given sample then is v x 76, where v is each 2-hour visit.

The table below shows basic statistics for the full dataset and for the subsample of data used in the current study. We use the subsample of data for the majority of experiments. The 6,945 stays are broken into 5,000 for training, 556 for validation, and 1,389 for test.

	Full Dataset	Sample Dataset
Total # patients	44,736	5,594
Total # ICU stays	56,384	6,945
Total # visits	3,858,779	462,289
# positive visits	80,115	9,544
Min/Max/Average length of stay (hours)	5 / 2,645 / 72.44	5 / 2,391 / 70.55
Min/Max/Average # of stays	1 / 29 / 1.26	1 / 17 / 1.24

3.4 Hyperparameters

The authors indicated that they had already performed hyperparameter tuning by grid search, so we use those same hyperparameters set out by the original paper and code.

Hyperparameter	Value
# of epochs	50
learning rate	0.001
batch size	128
hidden dimension	384
observation window (convolution kernel size)	10

3.5 Implementation

We use the author's existing code at https://github.com/v1xerunt/StageNet to implement the main StageNet model, and make further modifications to that code for ablation studies and improved reporting capabilities. Code for the current study can be found at https://github.com/joanna-regan/CS598_DL4H_StageNet.

3.6 Computational requirements

Code was predominantly run on Google Colab with GPU A100 and High-RAM enabled. We conducted initial data preprocessing and some experiments on a laptop with Windows 11, 11th Gen Intel(R) Core(TM) i7-1195G7, 2.92 GHz with 16GB RAM.

[1] https://physionet.org/content/mimiciii/1.4/

I tried to leverage cloud based GPU resources like Google Colab as much as possible, but encountered timeout errors where Google Colab was unable to load the entire training dataset. I tried saving the DataLoader instance locally using the torch.save() method but encountered local MemoryRuntime errors. I also tried subdividing the location of my training data into multiple, smaller, sub-directories but was still only able to load a portion of the dataset.

Initially, I began training the full dataset via CPU and found that training a single epoch took approximately 7 hours. I then decided to pare back to the subsample of 6,945 data points, which was the largest dataset I could load into Google Colab. When first training this subsample dataset locally with CPU, we found training times were long and inconsistent, as shown below:

Epoch1	Epoch2	Epoch3	Epoch4	Epoch5
1h:27m	31m	34m	33m	7h:20m
Epoch6	Epoch7	Epoch8	Epoch9	Epoch10
47m	40m	31m	40m	38m

I then ran the same subsampled dataset via GPU-enabled Google Colab and found training times to be much more consistent and we had an average epoch training time of 1m:14sec over 50 epochs.

4 Results

In Table 1, we show the performance of all models evaluated on both the full test dataset and the test subset as compared to the results reported by the original authors. We also show the results reported in the original paper for 2 baseline models, ON-LSTM (Shen et al., 2018) and Health-ATM (Ma et al.). These are used only for reference and no experiments were run in the current reproducibility study to verify these results.

For the Pre-Trained StageNet model, the authors provided a saved model in their github. We load this model, and report the performance when evaluating using the full test dataset (n=11,221) and when using the small subset (n=1,389).

For the remaining models (Reproduced StageNet, StageNet-I, and StageNet-II), we train a small subset of data (training set size = 5,000 and validation set size = 556) for 50 epochs using Google Colab with GPU A100 and high-RAM enabled. The model is then evaluated using the full test dataset (n=11,221) and using the small subset (n=1,389).

Following guidance of the original paper, we evaluate performance using the area under the receiver operating characteristic (AUROC), the area under the Precision-Recall curve (AUPRC), and the minimum of precision and sensitivity (min(Re, P+)). In order to determine which model is "best" during training, we looking for the model with the maxiumum AUPRC on the validation set.

4.1 Claim 1

We evaluate the claim that StageNet trained with MIMIC-III EHR data will achieve 10% higher AUPRC and min(Re, P+) than baseline models on decompensation risk prediction task.

We can confirm that when evaluating data against a pre-trained StageNet model provided by the original authors, we achieve 12% higher AUPRC than ON-LSTM and 17% higher AUPRC than Health-LSTM. We also achieve 13% higher min(Re, P+) than ON-LSTM and 20% higher than Health-LSTM. This performance is even better than that reported by the original authors.

However, we do not achieve the same stellar results for the models trained and/or evaluated on a smaller test set. In fact, not only do the reproduced/ablation models fail to achieve comparable performance to the results from the original paper, none of the remaining results outperform either baseline model on any of the reported AUPRC, AUROC, or min(Re, P+). The only exception is the min(Re, P+) for the pre-trained model evaluated on a subset of data, which is just 1% higher than ON-LSTM and 7% higher than Health-LSTM.

4.2 Claim 2

We evaluate the claim that reduced models StageNet-I and StageNet-II will still achieve higher AUPRC, AUROC, and min(Re, P+) than all baseline models.

Although the reduced models StageNet-I and StageNet-II do not outperform baselines models in the current study, we do see similar trends maintained from the original paper.

In the original paper, the AUPRC and AUROC for the main StageNet model was about 3-4% higher than that of the reduced models, and min(Re, P+) was within 1%. Our results show a similar trend, where AUPRC and AUROC for the main StageNet model are 2.8% - 3.6% higher than the worst reduced model, and min(Re, P+) is 1.7% higher than StageNet-II and StageNet-I is actually 7.8% higher than the main StageNet model. This trend

	AUPRC			AUROC			min(Re, P+)		
	Original	Full	Subset	Original	Full	Subset	Original	Full	Subset
Baseline1: ON-LSTM	0.304	—	—	0.895	—	—	0.343	—	—
Baseline2: Health-LSTM	0.291	—	—	0.897	—	—	0.325	—	—
Pre-Trained StageNet	0.323	0.341	0.289	0.903	0.909	0.890	0.372	0.390	0.347
Reproduced StageNet	0.323	0.228	0.206	0.903	0.874	0.842	0.372	0.292	0.280
Ablation1: StageNet-I	0.313	0.226	0.209	0.899	0.850	0.838	0.360	0.315	0.279
Ablation2: StageNet-II	0.311	0.220	0.211	0.897	0.872	0.844	0.358	0.287	0.280

Table 1: Performance comparison of baseline models against pre-trained StageNet model, reproduced StageNet model, and two different ablations StageNet-I and StageNet-II. The Pre-trained model was provided by the original authors, while the remaining models were trained as part of the current reproducibility study.

is most apparent when evaluating against the full test set. Results when evaluating with the subset test data are comparable across all three StageNet variations.

5 Discussion

If StageNet and both of its ablations were able to achieve higher evaluation metrics than baseline models, it would indicate that the novel components introduced by the original authors do in fact help with predicting a patient's risk of decompensation. Adding the stage-aware component to LSTM that can differentiate changes in disease stage based on recent versus old history, and the stage-adaptive component to the convolution model that can summarize stage information in a given observation window, both contribute to the overall success of the model.

Although I wasn't able to reproduce the same high levels of AUPRC, AUROC, or min(Re, P+) as were shown in the original paper, I think a few changes to my implementation would have yielded very comparable, if not better, results. The main limiting factor in my implementation was that I was only able to use a portion of the dataset to train my model. I was only able to load about 12% of the full dataset into Google Colab for training, and so the performance of my models suffered compared to the original paper. Additionally, this study was focused on reproducing the StageNet model and its ablations, but did not focus on recreating other state-of-the-art models used as a baseline. Consequently, I was not able to train baseline models using my small subset of data. It's not quite a fair comparison to juxtapose the baseline models trained on the full dataset to my reproduced models trained on a fraction of that dataset. Ideally, I would have either been able to train StageNet and its ablations with the full dataset, or train baseline models with the subsampled dataset in order to more accurately assess the reproducibility of the original paper.

Furthermore, I was able to achieve even higher AUPRC, AUROC, and min(Re, P+) when using the pre-trained model from the original paper and evaluating with the full test set. The original paper was published in 2020 and since then, the MIMIC-III database has added more samples. The original paper indicated they used the MIMIC-III dataset containing 41,902 ICU stays, whereas my full dataset contains 56,384 ICU stays. This additional data could possibly be used to train an even more powerful StageNet model.

5.1 What was easy

I found downloading and preparing the data for analysis was generally straightforward, especially given instructions earlier in the semester regarding how to obtain access to datasets on PhysioNet. Once I had the data available, the existing github repository [2] to create benchmarks for each task was well documented, easy to follow, and executed without major error.

It was also relatively easy to get the provided StageNet code running end to end. Though the github could have been documented a bit more, there was enough information to get something running. Unlike other papers I researched, they did not have extensive environment setup requirements and only required Pytorch be installed. Although they cannot provide the MIMIC-III data directly, they provided a few samples in the respective folders which I found very helpful to ensure that I had the data formatted correctly. I also found quite early on that they had a pre-built method to run the model using a small subset of data, which made

[2] https://github.com/YerevaNN/mimic3-benchmarks

my initial discovery much easier.

After spending time with the proposed algorithm in the paper and the code implementation, it was fairly easy to follow how each equation outlined in the paper was translated into code. This was also helpful when implementing the ablation studies because I knew which components were part of their novel approach and how to modify the code to test different aspects of their model.

5.2 What was difficult

Without prior experience using Google Colab, I found it difficult and often tedious trying to migrate training from my local machine. There were certain nuances in Google Colab that were unfamiliar to me and led to a lot of time spent trying to understand if there was a problem with my code or a problem with the environment. For example, I was never able to get my entire training dataset loaded into Google Colab after a lot of trial and error. Though I was able to load the entire dataset locally, the training times via CPU took 7h:00min:37sec and even when I only used a subset of the data, the training times were widely inconsistent (see Epoch5 as reported in Section 3.5 - over 7 hours for a fraction of the data!). This led to difficulties getting the appropriate compute resources in the right place - I could load data locally but since I only have CPU, the training times were prohibitively long. On the other hand, I had access to GPU via Google Colab but I could not get the full dataset loaded.

Further, this dataset is heavily imbalanced, with the positive class representing roughly 2% of the entire dataset. As such, attempting to train the model with smaller subsets of data was difficult because we would occassionally see a train, test, or validation set with all negative samples.

When running ablations, I struggled with the decision to implement from scratch or to dissect and modify the existing code. I chose to use the existing code as the skeleton architecture and then modify or remove components, in hopes that this would yield a more fair comparison between models. This led to extensive time spent dissecting each line of code to understand its contribution, but I came away with a much better understanding of the proposed StageNet model as well as LSTM in general since their existing method doesn't just call nn.LSTM() but steps through the underlying equation line by line.

5.3 Recommendations for reproducibility

I would suggest providing further documentation in the code to translate how each component compares to the equations provided in the paper.

I would also suggest providing implementation for ablation studies/reduced models of StageNet-I and StageNet-II. If available, I would suggest providing the code (or a link to the original author's code) for other baseline models referenced in the paper.

6 Communication with original authors

When I was originally planning to reproduce the MedFuse (Hayat et al., 2022) paper, I had reached out to their authors to get a sense of computational resources and training times because I was having extensive issues getting their code to run. However, no communication with the authors of StageNet was needed for the current study.

References

Junyi Gao, Cao Xiao, Yasha Wang, Wen Tang, Lucas M. Glass, and Jimeng Sun. 2020. Stagenet: Stage-aware neural networks for health risk prediction. *CoRR*, abs/2001.10054.

Amaral L. Glass L. Hausdorff J. Ivanov P. C. Mark R. ... Stanley H. E. Goldberger, A. 2000. Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *PhysioNet*, 101:e215–e220.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. 2019. Multitask learning and benchmarking with clinical time series data. *Scientific Data*, 6(1):96.

Nasir Hayat, Krzysztof J. Geras, and Farah E. Shamout. 2022. Medfuse: Multi-modal fusion with clinical time-series data and chest x-ray images.

Xi Zhang Fei Wang Anil K Jain Inci M Baytas, Cao Xiao and Jiayu Zhou. 2017. Patient subtyping via time-aware lstm networks. *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 65–74.

Pollard Tom J. Shen Lu Lehman Li-wei H. Feng Mengling Ghassemi Mohammad Moody Benjamin Szolovits Peter Anthony Celi Leo Mark Roger G. Johnson, Alistair E.W. 2016. Mimic-iii, a freely accessible critical care database.

Tengfei Ma, Cao Xiao, and Fei Wang. *Health-ATM: A Deep Architecture for Multifaceted Patient Health Record Representation and Risk Prediction*, pages 261–269.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron C. Courville. 2018. Ordered neurons: Integrating tree structures into recurrent neural networks. *CoRR*, abs/1810.09536.