CS598 DL4H Final Project Report Draft

Tony Mu and Xizi Liu
{tongm3, xiziliu2}@illinois.edu

Group ID: 157
Paper ID: 165
Presentation link: https://www.youtube.com
Code link: https://github.com/tonymuu/e2e_time_aware_attention

1 Introduction

Predicting readmission to the intensive care unit (ICU) and identifying patients at risk of readmission are becoming increasingly important in today's healthcare studies. Better prediction can not only help healthcare providers intervene early and potentially prevent ICU readmission, but also optimize healthcare resource allocations and therefore reduce healthcare costs.

To achieve more accurate predictions of readmission within 30 days of discharge from the ICU, and have a better identification of Patients-at-Risk, various approaches have been proposed to process time-series sampled at irregular time intervals, such as the diagnosis and procedure codes contained in EMRs. The paper "Benchmarking Deep Learning Architectures for Predicting Readmission to the ICU and Describing Patients-at-Risk" implemented and compared neural network architectures that have been proposed for processing longitudinal electronic medical records (EMR) data, and evaluated each considered algorithm against multiple criteria including average precision, AUROC, F1 score, sensitivity, and specificity. (1)

The insightful conclusions demonstrated in the paper can help healthcare providers better understand the intensive-care patients at increased risk of readmission and make the entire ICU readmission process more time-efficient and financially efficient.

2 Scope of reproducibility

Different techniques are utilized at different layers of the neural networks to build the models. This paper has implemented and compared thirteen neural network architectures for predicting readmission to the ICU by combining different techniques at each layer of the network.

2.1 Addressed claims from the original paper

There are three claims that our team is planning to test:

- Claim 1: The neural network architectures ODE + RNN, ODE + Attention, and ODE + RNN + Attention have similar results in ICU readmission prediction in terms of average precision, AUROC, and F1-score. Among the three models, ODE + RNN has the best result in average precision and AUROC, and ODE + RNN + Attention has the best result in F1-Score.
- Claim 2: Using neural ODEs to capture how the predictive relevance of recorded medical codes changes over time is a feasible approach.
- Claim 3: Models that include a recurrent component had slightly better accuracy than those relying only on attention mechanisms, and training attention-based networks is more efficient compared to training RNNs.

3 Methodology

To reproduce the study, we followed the method outlined in the paper and reused the authors' code to train the models. The data source would be the same as the original paper which is MIMIC-III.

3.1 Model descriptions

There are three neural network models we are planning to reproduce, ODE + RNN, ODE + Attention, and ODE + RNN + Attention. All the models first consume the data in the MIMIC-III tables such as diagnosis and procedure codes and map the data to corresponding "embeddings". After this stage, an embedding layer with additional neural ordinary differential equations (ODEs) is used to compute the time-aware code embeddings.

After obtaining the output from the embedding layer, for the first model ODE + RNN, embeddings are passed to RNN layers, and the final memory states are used for further processing. For model ODE + Attention, dot-product attention is then applied to the embeddings. And for model ODE + RNN + Attention, embeddings are passed to RNN layers as well as dot-product attention is applied to the RNN outputs.

Summary scores of diagnoses/procedures and medications/vital signs are then calculated, concatenated, and fed into the logistic regression layer which uses a fully connected layer with sigmoid activation.

3.2 Data descriptions

We used the Physionet's MIMIC-III dataset. We obtained the dataset by downloading it from the Physionet's website, after going through the required training and obtaining certificates. MIMIC-III is a large, freely-available database comprising deidentified health-related data associated with over forty thousand patients who stayed in critical care units of the Beth Israel Deaconess Medical Center between 2001 and 2012. The database includes information such as demographics, vital sign measurements made at the bedside (roughly 1 data point per hour), laboratory test results, procedures, medications, caregiver notes, imaging reports, and mortality (including post-hospital discharge)(2). There are 45298 patients, 25004496 diagnoses and procedures, and 17756816 charts and prescriptions in the dataset. Out of all patients, 5495 were readmitted and 39803 were not readmitted.

3.3 Hyperparameters

We inherited almost all hyperparameters from the original paper. The number of epochs we used is 80, the dropout rate we used is 0.5, and the batch size we used is 128. The only exception is the `bootstrap_samples` parameter. In the original implementation, the number of `bootstrap_samples` used is `100`. However, we found that using `100` samples is time-consuming on the hardware we used to do the experiments. Thus, we lowered the hyperparameter `bootstrap_samples` to `20`. This way, we were able to perform experiments in a more reasonable time frame.

3.4 Implementation

We used the existing code as a foundation to reproduce results in the paper. However, the existing code was slightly outdated, and we were not able to run the existing code and pipelines out-of-box. Thus, we also made modifications and additions to the existing code to do our own experiments. To commit these modifications and potentially propose these changes to the original authors, we forked the original GitHub repository and pushed our changes to the forked repository here https://github.com/tonymuu/e2e_time_aware_attention. We also implemented a customized model for ablation and the code was pushed to the GitHub link as well.

3.5 Computational requirements

We performed all of our experiments on the same virtual machine. The virtual machine has the following CPU: Intel(R) Xeon(R) CPU E5-2673 v3 @ 2.40GHz 4 Core, and 16GB of RAM. There is no GPU on the machine. We ran a total of three tests and found the following CPU computation times for each test:

- For the ODE + RNN model, the number of training epochs is 80. We found CPU time to be 2893.636 seconds, with a 95% confidence interval between 2827.862 seconds and 2959.41 seconds. The standard deviation is 293.708 seconds.

- For the ODE + Attention model, the number of training epochs is 80. We found CPU time to be 181.48 seconds, with a 95% confidence interval between 181.186 seconds and 181.774 seconds. The standard deviation is 1.312 seconds.

- For ODE + RNN + Attention, the number of training epochs is 80. We found CPU time to be 3394.986 seconds, with a 95% confidence interval between 3388.251 seconds and 3401.721 seconds. The standard deviation is 30.074 seconds.

In the proposal, we determined that existing computational resources will be sufficient to perform the experiments. While this is the case, we found these experiments more time-consuming than we originally estimated. Thus, we had to reduce the sample size when reproducing experiments. Moreover, although we did not include an estimated

computational time in the proposal, we underestimated the actual computational resources needed to perform these experiments.

4 Results

After training and evaluating the models, statistics have been obtained as shown in the following tables.

Model	Average Precision
ODE + RNN	0.339 [0.318,0.36]
ODE + Attention	0.297 [0.275,0.319]
ODE + RNN + Attention	0.319 [0.299,0.339]

Table 1: Average Precision for the reproduced models.

Model	AUROC
ODE + RNN	0.741 [0.734,0.748]
ODE + Attention	0.716 [0.708,0.725]
ODE + RNN + Attention	0.739 [0.732,0.747]

Table 2: AUROC for the reproduced models.

Model	F1-Score
ODE + RNN	0.376 [0.363,0.389]
ODE + Attention	0.334 [0.321,0.347]
ODE + RNN + Attention	0.379 [0.366,0.392]

Table 3: F1-Score for the reproduced models.

As demonstrated in the tables, for the algorithm ODE + RNN, we reproduced the average precision to 2.417% of the reported value, the AUROC to 0.271% of the reported value, and the F1-Score to 1.075% of the reported value.

For the algorithm ODE + Attention, we reproduced the average precision to 1.020% of the reported value, the AUROC to 0.139% of the reported value, and the F1-Score to 0.300% of the reported value.

And for the algorithm ODE + RNN + Attention, we reproduced the average precision to 1.592% of the reported value, the AUROC to the same as the reported value with a 33.333% wider confidence interval, and the F1-Score to 0.798% of the reported value.

4.1 Result 1 - Claim 1

From the results displayed above, it is clear that models: ODE + RNN, ODE + Attention, and ODE + RNN + Attention have similar performance in predicting the risk of readmission within 30 days of discharge from the ICU using the MIMIC-III data set. The model ODE + RNN has the best average precision (0.339 [0.318,0.36]) and best AUROC (0.741 [0.734,0.748]), and the model ODE + RNN + Attention has the best F1-Score (0.379 [0.366,0.392]). This result upholds the paper's conclusion as well as claim 1.

4.2 Result 2 - Claim 2

To verify if neural ODEs are feasible for processing recorded medical codes, we reproduced three more models which use MCE instead of ODE. The results are shown below.

Model	Average Precision
MCE + RNN	0.305 [0.285,0.324]
MCE + Attention	0.273 [0.252,0.295]
MCE + RNN + Attention	0.323 [0.302,0.344]

Table 4: Average Precision for the reproduced MCE models.

Model	AUROC
MCE + RNN	0.727 [0.719,0.735]
MCE + Attention	0.69 [0.682,0.697]
MCE + RNN + Attention	0.737 [0.729,0.745]

Table 5: AUROC for the reproduced MCE models.

Model	F1-Score
MCE + RNN	0.365 [0.353,0.378]
MCE + Attention	0.315 [0.305,0.325]
MCE + RNN + Attention	0.378 [0.367,0.389]

Table 6: F1-Score for the reproduced MCE models.

From the results displayed in Tables 1, 2, 3, 4, 5, and 6, in terms of average precision, models with only ODE + RNN and ODE + Attention outperforms MCE + RNN and MCE + Attention respectively, and ODE + RNN + Attention and MCE + RNN + Attention share very similar results. In terms of AUROC and F1-Score, models using ODEs have a better performance than models using MCE.

All three models using neural ODEs at the embedding layer to process time-series sampled at irregular time intervals succeeded to provide reasonable results when predicting ICU readmission,

which means neural ODE is a novel method to model how the predictive relevance of medical codes changes over time. This result upholds the paper's conclusion as well as claim 2.

4.3 Result 3 - Claim 3

From section 3.5, we can see that training model ODE + RNN requires a CPU time of 2893.636 seconds on average, whereas training model ODE + Attention requires a CPU time of 181.48 seconds on average which is significantly less. From Tables 1,2 and 3, model ODE + RNN marginally outperforms model ODE + Attention in terms of average precision, AUROC, and F1-Score. This result upholds the paper's conclusion as well as claim 3.

4.4 Additional Experiments

In order to explore the feasibility of using ODE and MCE and to better understand the effect of the dropout rate, there are two additional experiments we performed.

4.4.1 Additional Experiment 1

Since training models with only attention layers is more efficient than training models with RNNs, we implemented an additional model MCE + ODE + Attention to explore the overall performance with both ODE and MCE used to process the time-series sampled at irregular intervals. We compared the result of this self-implemented model with the result of model ODE + Attention. Since training is very time-consuming, we reduced the number of epochs to 5. The results for these two models are shown in Tables 7 and 8.

Model	MCE + ODE + Attention
Average Precision	0.277 [0.254,0.3]
AUROC	0.676 [0.667,0.685]
F1-Score	0.309 [0.296,0.322]

Table 7: Performance for MCE + ODE + Attention (5 Epochs).

Model	ODE + Attention
Average Precision	0.274 [0.251,0.296]
AUROC	0.678 [0.669,0.687]
F1-Score	0.312 [0.301,0.324]

Table 8: Performance for ODE + Attention (5 Epochs).

From the results, we can see that in terms of average precision, MCE + ODE + Attention performs slightly better than ODE + Attention, whereas in terms of AUROC and F1-Score, ODE + Attention outperforms MCE + ODE + Attention. Training MCE + ODE + Attention (5 epochs) used 1 hour and 9 minutes, and training ODE + Attention (5 epochs) used 1 hour and 6 minutes, the training time required for these two models is pretty close. Therefore, we concluded that adding one extra layer of MCE for processing recorded medical codes is not necessary, and using either ODE or MCE should be sufficient.

4.4.2 Additional Experiment 2

Another additional experiment was done by modifying the dropout rate from 0.5 to 0.75 to understand the effect of changing the dropout rate parameter. To save the training time, we trained model ODE + Attention and reduced the epoch number from 80 to 40.

Model	ODE + Attention
Average Precision	0.293 [0.27,0.316]
AUROC	0.702 [0.692,0.711]
F1-Score	0.329 [0.316,0.341]

Table 9: Performance for ODE + Attention (0.75 Dropout Rate & 40 Epochs).

As shown in Table 9, comparing the results with the one from Table 1, 2, and 3, it is obvious that they have similar statistics. Therefore, we concluded that it is likely that increasing the dropout rate has little impact on the average precision, AUROC and F1-Score. But final conclusion should be made based on more thorough experiments.

5 Discussion

We conducted experiments to compare our results with some of the claims made by the original authors, and we successfully reproduced all the claims we tested. From the results we obtained, we concluded that when there is a need to process times-series samples at irregular intervals, aside from using MCE which is one of the most popular options nowadays, using neural ODEs can also be a feasible and innovative choice. The performance difference between these two approaches is neglectable. Additionally, for the three models ODE + RNN, ODE + Attention, and ODE + RNN + Attention, if average precision is more important, ODE + RNN will be the best model to use, and if classifier quality has higher priority, either ODE +

RNN or ODE + RNN + Attention can be the candidates to choose from. Lastly, since the overall performance of models that include RNNs is very close to the overall performance of models that rely only on attention mechanisms, whereas training RNNs is more time-consuming than training attention-based networks, using networks that rely only on attention layers is recommended.

However, certain limitations hindered us from conducting further experiments to validate our own developed models. Despite these limitations, the authors presented their ideas effectively, theorized their claims well, and provided code that facilitated reproducibility. However, we were unable to reproduce all the claims presented in the paper. The main obstacle was our limited computational resources, as our machine lacked sufficient power to conduct extensive experiments. For instance, running a training pipeline with default hyperparameters (num_epochs = 80) on our virtual machine would require nearly 10 hours. Another factor that complicated the reproduction process was unclear or missing instructions. The cloned repository could not be executed without additional troubleshooting and debugging to address error messages and achieve an initial successful run.

5.1 What was easy

Reproducing the results turned out to be quite smooth overall. We easily used the pretrained model, so we didn't need more training after successfully running the process. This saved us lots of time and effort, letting us focus on the important analysis and evaluation stages. Additionally, preprocessing the data was quite straightforward since the accompanying code was carefully designed, showing high reliability and stability. We hardly encountered any bugs, making it easy to convert and clean the raw data for our deep learning models. Furthermore, the training and testing pipeline seamlessly fit into our workflow, needing only minor fixes to ensure everything worked smoothly. This well-structured pipeline not only helped us evaluate our models effectively but also provided a trustworthy baseline for comparing them with the models discussed in the paper. By reusing the pipeline, we smoothly integrated our own models and achieved consistent, dependable results, ultimately saving valuable time and resources.

5.2 What was difficult

Reproducing the experiment results outlined in the research paper presented several challenges. Firstly, we encountered difficulties due to the lack of clear instructions in the original code repository. The absence of explicit guidance meant that we had to meticulously read through the repository ourselves to understand the purpose and functionality of each script file. This process was time-consuming and required additional effort to debug and troubleshoot any encountered error messages. Despite these obstacles, we were able to navigate through the repository and gain a comprehensive understanding of the code.

Furthermore, another obstacle arose from the absence of dependency specifications, specifically the absence of a requirements.txt file for Python. This omission made it challenging to determine the specific versions of the required packages. Consequently, we encountered backward-compatibility errors when attempting to run the code with different package versions. These errors could have been avoided with clear dependency specifications, saving us time and effort in resolving compatibility issues.

Additionally, reproducing the experiment results was hindered by the demanding computational requirements of the dataset and default hyperparameters, particularly the num_epochs value. Unfortunately, our available computational resources were not sufficiently powerful to handle the substantial computational workload. As a result, training the models became highly inefficient and time-consuming, with each iteration taking approximately 20 hours to complete. This significantly impeded our ability to develop and iterate on our own model, limiting our progress and hindering experimentation.

5.3 Recommendations for reproducibility

To address the encountered difficulties during the reproduction process, we would like to offer the following recommendations to the authors. Firstly, we suggest creating clear and concise instructions that provide step-by-step guidance on setting up the environment and executing the pipelines. This will greatly assist future researchers in understanding and replicating the experiment procedures effectively. Secondly, it would be beneficial to implement a dependency specification, such as a requirements.txt file, that precisely lists all the required

packages and their compatible versions. This will ensure that researchers can easily install the necessary dependencies without facing compatibility issues. Lastly, we recommend providing detailed information about the computational resources utilized and the time required to produce the experiment results. This transparency will enable researchers to better plan and allocate sufficient computational power for reproducing the experiments. By incorporating these recommendations, future replication attempts can be facilitated, contributing to the overall reproducibility and advancement of the research.

References

[1] Barbieri, S., Kemp, J., Perez-Concha, O., Kotwal, S., Gallagher, M., Ritchie, A., & Jorm, L. (2020). Benchmarking Deep Learning Architectures for predicting readmission to the ICU and describing patients-at-risk. Scientific Reports, 10(1). https://www.nature.com/articles/s41598-020-58053-z

[2] Alistair Johnson (2016). MIMIC-III Clinical Database. https://physionet.org/content/mimiciii/1.4