CSE 6250 Project Report: Time-Aware and Co-Occurrence-Aware Network for Medical Prediction Reproducibility

Joe Laniado
Georgia Institute of Technology
jlaniado3@gatech.edu

Yumei Wang
Georgia Institute of Technology
ywang4068@gatech.edu

Abstract

The Big Data for Healthcare reproducibility challenge consists of selecting from a pool of papers to apply the knowledge learned throughout the semester on a real life healthcare application. In Interpretable time-aware and co-occurrence-aware network for medical prediction [7] they leverage Electronic Health Records to propose a deep learning approach that takes into consideration the time and co-occurrence factor for cardiovascular diseases on each patient. We implement a simplified approach using knowledge practiced and learned throughout the semester to achieve comparable results on multiple classification tasks specified in the paper.

1. Introduction

Interpretable time-aware and co-occurrence-aware network for medical prediction [7] proposes a novel approach for predictive modeling leveraging Electronic Health Records (EHRs) and deep learning methodologies. It initially emphasizes how current state of the art approaches take advantage of the hierarchical structure for EHRs by either modeling the sequential nature of admissions for each patient or by trying to make sense of the co-occurrence of diseases inside each individual admission, usually inferring causation, mortality, or further complications. Although, no general approach has been proposed that both approaches can be used to model the data simultaneously due to the pre-processing and context required of the data for each one. For time aware methods the data needs to be presented on a sequential manner, while for co-occurrence modeling it needs to be in the form of a bipartite graph.

Another point mentioned was how difficult it is to interpret deep learning models due to the black box nature of each algorithm and neural networks. This poses a challenge for non-specialists to gain a more accurate picture of each patient and it hinders the ability for doctors to give more personalized diagnostics for each patient.

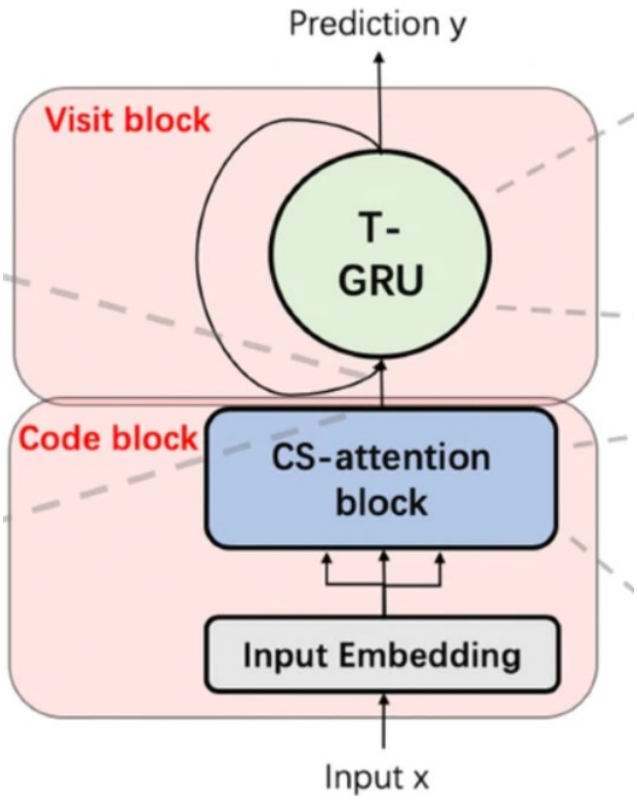

Figure 1. Original Structure of TCoN Network.

To try and tackle these issues the paper defines EHRs as **the hierarchical co-occurrence sequence** and it proposes a new network architecture called **TCoN** (Time-Aware Co-Occurrence Network). The network structure consists of an initial embedding layer that then passes the data to a Co-Occurrence self attention block, this allows the network to efficiently model the co-occurrence relationship between each code both at each visit and between admissions, and provides an interpretable path for the user to gain further insight on the relationships. The results are then connected to a Time-aware Gated Recurrent Unit (T-GRU), to capture

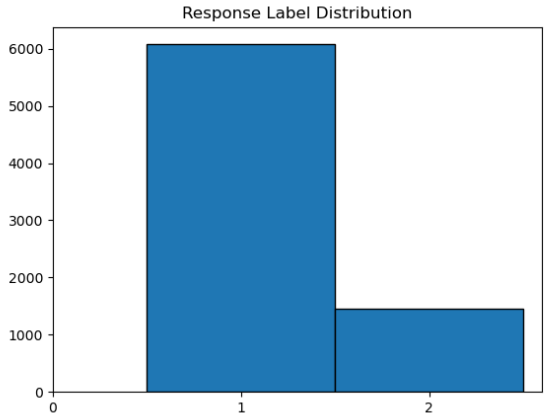

Figure 2. Mortality Response Label Distribution

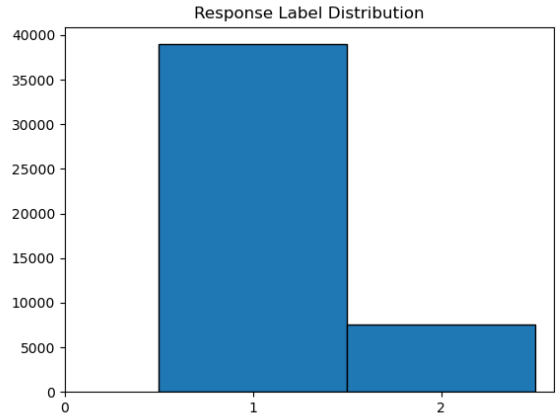

Figure 3. Readmission Response Label Distribution

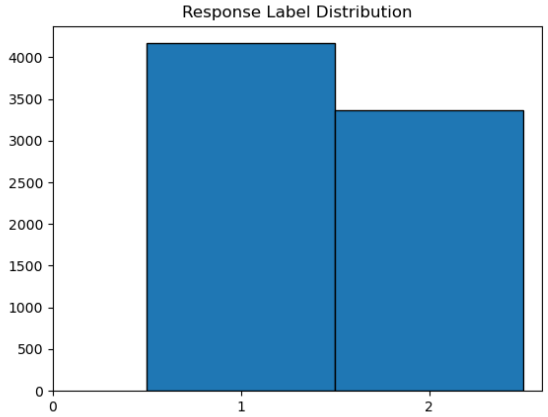

Figure 4. Heart Failure Response Label Distribution

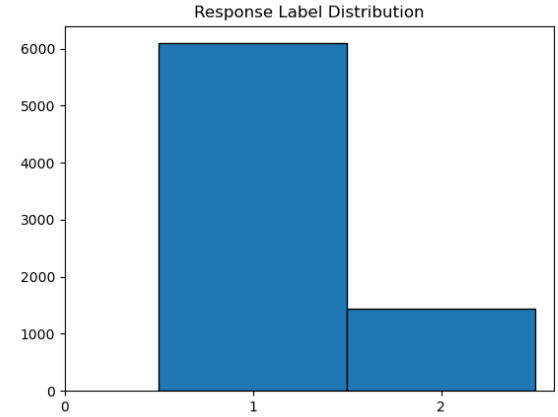

Figure 5. Sepsis Response Label Distribution

Figure 6. Response Label Distribution for Each Dataset.

the sequential time-aware relationship of each distinct admission for each patient. Both steps are then joined by an attention connection.

Finally, the newly developed network is used on 4 different predictive tasks: mortality prediction, readmission prediction, disease prediction, and next diagnoses prediction. It is then compared to state of the art models on performance and accuracy.

2. Scope of reproducibility

Our implementation will attempt to recreate the proposed network and evaluate on the 4 binary classification tasks proposed in the paper:

1. Mortality prediction: To predict if the patient will die in the hospital.
2. Readmission prediction: To predict if the patient will be hospitalized again.
3. Heart Failure Diagnostic: early diagnosis of disease.
4. Sepsis Diagnostic: early diagnosis of disease.

To eliminate complexity in reproducing the project, the multi-label next diagnoses task will be saved for future work. The results will be evaluated using precision, recall, accuracy, and AUC-ROC. We will compare our metric to those listed on the original findings for the time-aware (GRU [2] and T-LSTM [1]), Co-Occurrence methods (Med2Vec [1] and Dipole [5]), and the original TCoN network. Initial results will be presented using the original parameters used in the paper, but for our implementation some hyper-parameter tuning will be tried to achieve a better score using our network. We do this because either our

pre-processing or the way we simplified the network will cause the network to not perform the same way that the original TCoN did. Finally, we will determine if any changes to the original structure could achieve a better outcome and any challenges faced when trying to reproduce the original results.

3. Methodology

3.1. Model descriptions

Our model takes a more simplified interpretation of the original TCoN network. It follows the same basic architecture of an initial embedding layer to reduce the non-sparsity of the data, a multi-head self attention to capture the co-occurrence relationship between each disease, and finally a time-aware GRU network to capture the sequential relationship of each visit.

The embedding layer takes the total number of features and reduces it to 200 to reduce sparsity, it is then passed into a ReLu activation function as part of the fine-tuning process for the self attention block, finally a second linear layer is applied to reduce the features to 100. For the self attention step, we use a PyTorch multi-head attention layer with 2 heads, a local head for capturing the co-occurrence inside each admission and a global head to capture the relationship for codes between admissions. An attention connection between the self-attention block and the time aware GRU layer was proposed in the initial paper but for our code we decided not implemented to reduce complexity and avoid extra overfitting. The GRU block consist of a single layer which outputs 2 distinct features that serve as the binary classification output for the specific task at hand. The model is designed to handle a variable length input by selecting the latest non-padded output as the classification assignment for each patient.

We used cross entropy loss as our loss function for training the network. The network was trained using the Adam Optimizer [4] with a $lr = 0.001$, $\beta_1 = 0.9$, and $\beta_2 = 0.999$; the same as in the original paper. The original paper pre trains the network using an auto-encoder network. The parameters of the encoder layer are used as the initial parameters of TCoN when training by the prediction objective, but for our implementation we tried to get similar results without doings this step.

3.2. Data descriptions

All of the data was sourced from the same origin as the original paper: the MIMIC-III clinical database [3]. A large and freely-available database made up of health related information for more than forty thousands deidentified patients staying in critical care units from 2001-2012 at the Beth Israel Deaconess Medical Center. Three tables were used from the database: Admission, diagnoses-icd and patients. In preparation to evaluate each task we extracted records with more than one visit from the admission table and joined with the diagnoses-icd table. The new dataset comprises 19,993 hospital admissions of 7537 patients and 260,282 diagnoses with 4,893 unique codes defined by the International Classification of Diseases-9 version (ICD-9). This data was further processed and filtered to extract each specific dataset used for each one of our prediction tasks.

3.2.1 Mortality dataset

To accurately predict if a patient would die give a their Electronic Health Record we needed to create a mortality label given a series of diagnostics for each person. To do this we followed the procedures taught on the homework and used the DEATHTIME variable in the admission table. If DEATHTIME was null then our mortality label would be a 0, otherwise, a 1.

3.2.2 Sepsis dataset

Following the latest sepsis 3.0 definition [6], it can be defined as a life-threatening organ dysfunction caused by a dysregulated host response to infection. The ICD9 Code for this disease is 995.92 and we extracted 1444 patients from the original data-set that presented this disease and whose SOFA was greater than or equal to 2. For the prediction task we made our observation window all the different diagnoses before the patient presented the sepsis code and made our response variable a 1 if the patient presented sepsis and a 0 if they didn't for their next disease.

3.2.3 Heart failure dataset

To determine if a patient presented a diagnosis of heart failure we followed the guidance from the original paper and extracted all of those patients that had a diagnosis with an ICD9 code that started with 428.x. This resulted on 3370 individuals out of the original processed data. Our observation window was all the diseases before the diagnosis of heart failure and we made our response variable a 1 if the patient presented the disease and a 0 otherwise, the same methodology used on the sepsis dataset.

An important factor to note for both the Heart failure and sepsis data, is that both datasets could be classified as imbalanced. There are few positive cases and more negative cases. This will have to be taken into account when creating and evaluating our model to achieve a more accurate result.

3.2.4 Readmission dataset

This specific dataset represented a challenge as the original paper gave no instructions or explanation of what they would classify as a readmission given a certain number of

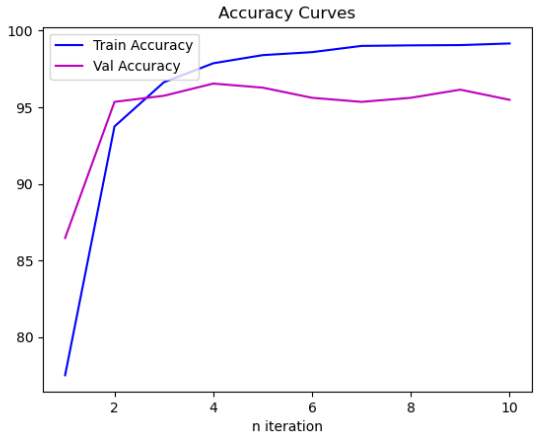

Figure 7. Accuracy curve

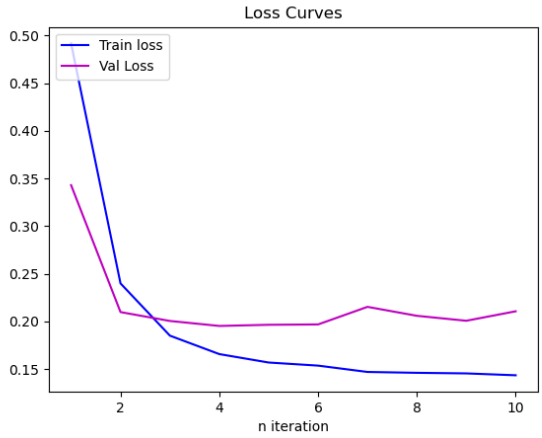

Figure 8. Loss Curve

Figure 9. Example Learning Curves After Hyper-parameter Tuning.

Method	Mortality		Readmission		Sepsis		Heart Failure	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
GRU	0.7902	0.7400	0.7023	0.6713	0.6202	0.6063	0.6525	0.6187
Med2Vec	0.8025	0.7950	0.7125	0.6833	0.8211	0.7943	0.7225	0.7101
Dipole	0.8133	0.8103	0.7341	0.7243	0.8001	0.7823	0.7067	0.6923
TLSTM	0.7893	0.7392	0.7256	0.7023	0.6432	0.6189	0.7432	0.6033
TCoN	0.8224	0.8134	0.7403	0.7278	0.8433	0.8233	0.7698	0.7313
TCoN BDH	0.7605	0.4737	0.6955	0.2932	0.9506	0.8758	0.8321	0.7747

Table 1. Results of original and our binary classification.

visits. For our implementation we decided that given a certain observation window if the patient happens to have a further visit it would be classified as a 1, if the patient never came back then it would be a 0. Due to the vagueness of this steps preprocessing our results might differ from those of the original paper. Most of the data needed to filter this data were based on the time variables found on the admission tables.

3.2.5 Distribution of Data

The same distribution of data was used for all 4 of the prediction tasks, the same used to train and evaluate the original TCoN network. All 4 datasets were split into train, validation and test sets in the ratio of 0.75:0.1:0.15 respectively. A randomized sample was taken for this distribution making sure that the imbalanced data did not cause any issues for training the model.

3.3. Computational Implementation

Most preprocessing was done using python and google colab. Pytorch was the decided library for implementing most of the code for the actual network. For initial experiments our TCoN implementation does not have any dropout rate following the guidance from the original paper and is trained for 10 epochs for each task with a batch size of 32. The original TCoN network has a computational complexity of $O(n * d^2)$, where d is the representation dimension and n is the sequence length. We determine that even with the slight changes of our network, our TCoN would have a similar complexity due to using the same overall architecture and data. Most of the training is done using our local CPUs at the moment making it take around 10 minutes for the network to learn. GPUs using google collab could be tried to be incorporated in the future for faster hyper-parameter tuning and cross-validation evaluations for more comprehensive results.

3.4. Code

The original code for the paper implementation was mainly developed using TensorFlow and it's publicly available at https://github.com/SCXsunchenxi/MTGRU. Our initial attempts to try and recreate the original work proved difficult as each specific cell in the network was made form scratch and the scripts lacked any documentation of any process or variables throughout the code. Furthermore, a deep understanding of the mathematics and intricate measures of each layer was needed to reproduce the original

Table 2. Results of Hyper-parameter Tuning on TCoN BDH

Method	Mortality		Readmission		Sepsis		Heart Failure	
	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC	ROC-AUC	PR-AUC
Lr = 0.001, Drp = 0	0.7605	0.4737	0.6955	0.2932	0.9506	0.8758	0.8321	0.7747
Lr = 0.005, Drp = 0	0.5224	0.2158						
Lr = 0.0005, Drp = 0	0.7791	0.5162	0.6982	0.2909	0.9696	0.9105	0.8442	0.7805
Lr = 0.0005, Drp = 0.1	0.8081	0.5438	0.6987	0.2870	0.9748	0.9132	0.8481	0.8227
Lr = 0.0005, Drp = 0.15	0.8077	0.5384	0.6978	0.2879	0.9764	0.9161	0.8465	0.7880
Lr = 0.0002 , Drp = 0.1	0.8094	0.5203	0.6982	0.2878	0.9765	0.9096	0.8365	0.7659

methodology proposed in the paper.

Due to all of this constraints we decided to perform a simplified version of the TCoN network in hopes of achieving similar results with a more general approach. Most of our code takes a similar approach to those used during the last couple of homeworks. This applies to pre-processing, training, and evaluation of our network. The intent was to use the skills learned in class to reproduce the complex methodology used on the original codebase. All of our code can be found at https://github.com/joelaniado/CS6250_Project.

4. Results

The original results for each binary classification task can be found in 1. The last row of the table displays the results for our implementation. Our network seemed to have performed better than the original TCoN when predicting disease, but worse when predicting mortality or readmission. This could be due to a number of factors: variation on the preprocessing, simplification of the network itself, or specific hyper parameters when building and training the network. Another issue to consider is that the validation learning curve for our network seems to learn very slowly compared to the training curve as shown in 9. Data preprocessing and validation of methodology were performed but no obvious reason for this behaviour were found. Overall initial results seemed satisfactory for the task and the level of complexity of the original TCoN.

Hyper-parameter tuning was also implemented on the learning rate and dropout on the multi-head attention block with hopes of improving initial results. Table 2 provides a detailed description for each task given certain values for dropout and learning rate. These two were chosen due to initial results showing over-fitting of our network on different tasks. Given the poor performance of increasing the learning rate, it was only attempted for a single task and not reproduced for the others. Tuning the parameters resulted in a slight increase in AUC across the board for each task. A slight dropout introduction as well as lowering the learning rate proved to be the most efficient strategy.

5. Discussion

Our experiments showed that the paper could be reproduced but it requires a deeper understanding of deep learning concepts and processes to truly grasp their methodology and manipulate the different weights, biases, and data to achieve a similar result with a proper interpretation path. Our implementation tried to reproduce the TCoN network using only knowledge learned from the homeworks and using the pre-built layers from the PyTorch package to create and train our network to the best of our ability. In contrast the original paper built every cell from scratch calculating the outputs by multiplying the weights and biases with the data directly using TensorFlow. This explains why our scores were much lower than the original network, and goes to show that unless you have a deep level of understanding of each deep learning layer, the proper data preprocessing, and a higher level of customization of the network; it will be really hard to reproduce the original results.

While the overall idea of the paper is easily understood, it lacks serious documentation on how they pre-processed their data, specific network configurations, and how they organized their code. The Github repository provided lacks any comments or descriptive naming convention for each file making it really hard to even know what each script does.

The paper itself provides a great tool for medical professionals that could be adapted to many predictive modeling tasks as well as provided deeper insight on personalized healthcare for each patient, but for their work to be easily reproduced, further documentation is needed on network specific configurations, data preprocessing, and the overall code implementation of their approach.

While our implementation did not meet the high standards from the original paper, if more time had been given and we had a better understanding of deep learning methodologies, our results could have been similar to those achieved by the TCoN Network.

6. Video link

Video link: https://youtu.be/XfYD5BIW51g

References

[1] Inci M Baytas, Cao Xiao, Xi Zhang, Fei Wang, Anil K Jain, and Jiayu Zhou. Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, New York, NY, USA, Aug. 2017. ACM. 2

[2] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014. 2

[3] Alistair Johnson, Tom Pollard, and Roger Mark. MIMIC-III clinical database. 2023. 3

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 3

[5] Fenglong Ma, Radha Chitta, Jing Zhou, Quanzeng You, Tong Sun, and Jing Gao. Dipole. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2017. 2

[6] Mervyn Singer, Clifford S. Deutschman, Christopher Warren Seymour, Manu Shankar-Hari, Djillali Annane, Michael Bauer, Rinaldo Bellomo, Gordon R. Bernard, Jean-Daniel Chiche, Craig M. Coopersmith, Richard S. Hotchkiss, Mitchell M. Levy, John C. Marshall, Greg S. Martin, Steven M. Opal, Gordon D. Rubenfeld, Tom van der Poll, Jean-Louis Vincent, and Derek C. Angus. The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3). *JAMA*, 315(8):801–810, 02 2016. 3

[7] Chenxi Sun, Hongna Dui, and Hongyan Li. Interpretable time-aware and co-occurrence-aware network for medical prediction. *BMC Medical Informatics and Decision Making*, 21(1):305, Nov 2021. 1