Reproducibility Report Draft for CS598 DL4H in Spring 2023

Hanyin Wang and Chang Guo
{hanyinw2, changg3}@illinois.edu

Group ID: 22
Paper ID: 158

1 Introduction

It is a central task in hospital operation and resources planning to reliably predict the length of stay for hospitalized patients, especially for those critically ill patients staying in the Intensive Care Unit (ICU). Our selected paper aims to build a novel and reliable deep-learning predictive model for the length of stay in the ICU(Rocheteau et al., 2021).

Historically, the two most popular models used for the length of stay prediction are LSTMs and Transformers due to the centrality of time series in the EHR (Sheikhalishahi et al., 2019; Song et al., 2018). More recently Temporal Convolutional Networks (TCN) were developed as a variation of CNN to handle sequential data (Bai et al., 2018). Our selected paper described a novel Temporal Pointwise Convolution (TPC) model, which is based on the combination of temporal convolution networks and pointwise (1x1) convolution to predict ICU length of stay. The TPC model is specifically designed to handle common challenges with EHR, such as skewness, irregular sampling, and missing data. In the paper, the TPC model significantly outperforms the commonly used LSTM and Transformer models for ICU length of stay prediction.

2 Scope of reproducibility

We aim to reproduce the finding that the novel TPC model based on temporal convolution networks combining with pointwise convolution will have lower mean absolute deviation (MAD) and mean absolute percentage error (MAPE) in predicting ICU length of stay than LSTM and transformer models trained on eICU database.

2.1 Addressed claims from the original paper

- The TPC model significantly outperforms the commonly used LSTM and Transformer models for ICU length of stay prediction by margins of 18-68%.
- The use of mean-squared logarithmic error (MSLE) loss function instead of commonly used mean squared error (MSE) loss will lead to better model performance, as the former handles positively-skewed labels more naturally.
- The data processing pipeline developed by the authors for the eICU and MIMIC-IV databases could mitigate the common problems of sparsity and missing data in the EHR and the approach is generalizable to other EHR databases.

3 Methodology

3.1 Model descriptions

Initially, for every timepoint *t*, there are two channels per time series feature: F feature values $x_t^{'} \in \mathbb{R}^{F\times 1}$ and their corresponding decay indicators $x_t^{''} \in \mathbb{R}^{F\times 1}$. The decay indicators tell the model how recently the observation $x_t^{'}$ was recorded. As we pass through the layers of model, we repeatedly extract trends and inter-feature relationships.

Author used stacked TCNs to extract temporal trends in the data. TCNs are a subclass of CNN that evolve over the time dimension (Figure 2a). They operate on two key principles: the output is the same length as the input, and there can be no leakage of data from the future(Bai et al., 2018). The paper made a unique adjustment to not allow weights sharing across features, thus weight sharing is only across time. The philosophy is that the features from EHR differ sufficiently in their temporal characteristics to warrant specialized processing.

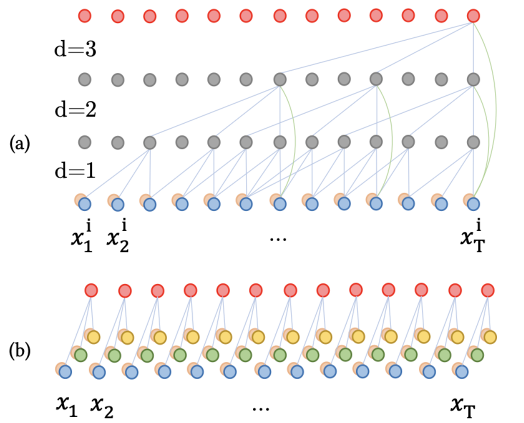

Figure 2: (a) Temporal convolution with skip connections (green lines). Each time series, *i* (blue dots) and their decay indicators (pale orange dots) are processed with independent parameters. (b) Pointwise convolution. There is no information sharing across time, only across features (blue, green, yellow dots).

Pointwise convolution is traditionally used to reduce the channel dimension when processing images. It can be conceptualised as a fully connected layer, applied separately to each timepoint (Figure 2b). Contrary to TCNs, in the paper weights are shared across features in pointwise convolution to obtain interaction features, while there is no information transfer across time.

Finally, the TCP model combines TCNs and pointwise convolution in parallel (figure 3). In a TPC layer the temporal output is combined with the skip connections first, then concatenated with the pointwise output after latter has been broadcasted. The full model has N TPC layers stacked sequentially. After N layers, the output h_t^N is combined with static features $s \in \mathbb{R}^{S\times 1}$, and a diagnosis embedding $d^* \in \mathbb{R}^{D\times 1}$. Two pointwise layers are then applied to obtain the final predictions. The model also enables flexible selection of temporal receptive field sizes (independently for each feature) because of the skip connections. Batch normalisation and dropout are used throughout to regularize the model.

Of interest, the author selected mean squared log error (MSLE) as the loss function in training instead of the commonly-used mean squared error (MSE) loss. This choice aims to address the positive skew in LOS.

3.2 Data descriptions

The paper used the eICU Collaborative Research Database v2.0 and MIMIC-IV v0.4 Database, both available through PhysioNet. We selelcted eICU

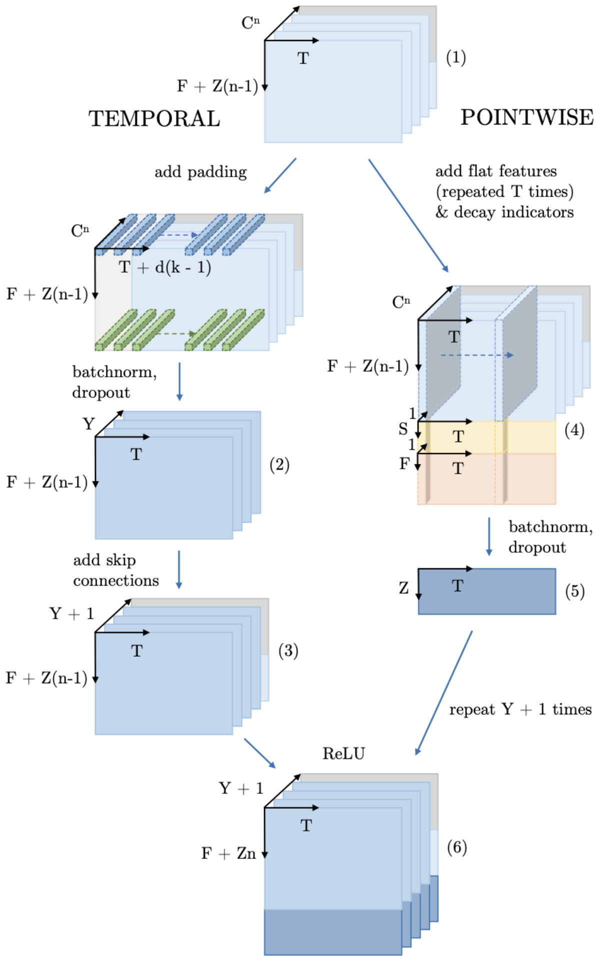

Figure 3: The n_{th} TPC layer. Left-sided padding (off-white) is added to the temporal side before each feature is processed independently. On the pointwise side, flat features (yellow) and decay indicators (orange) are added before each convolution.

data for the purpose of replication, and statistic overview of the eICU data is listed in table 1.

eICU data are first downloaded and build into postgreSQL database following steps in: https://github.com/MIT-LCP/eicu-code/tree/master/build-db/postgres. A customized SQL file provided by the paper is subsequently run to generate all input files. Of note, due to limitation of computation resources, we only selected eICU database and utilized 50% of hospital stays in the training of model. We find this compromise to be acceptable as the performance superiority of TPC model is already evident when using 50% training set as demonstrated in table 12 of original paper.

3.3 Implementation

We reused the existing codes by the author and tailored it to be run on Google Colab (https://github.com/EmmaRocheteau/TPC-LoS-

Table 1: eICU Cohort Summaries

	eICU
Number of patients	118,535
Train	82,973
Validation	17,781
Test	17,781
Number of stays	146,671
Train	102,749
Validation	22,033
Test	21,889
Gender (% male)	54.1%
Age (mean)	63.1
LoS (mean)	3.01
LoS (median)	1.82
Number of input features	104
Time series	87
Static	17

prediction). We made several adjustment to fix bugs due to package and version incompatibility. We also write codes for the models to be run on 50% of training data set. We write new codes for the new experiments not designed in the paper.

3.4 Computational requirements

The eICU database preprocessing was performed on a M1 MacBook pro but unfortunately had to be done under a x86 architecture due to package compatibility issue, therefore significantly limited CPU performance. The whole data preprocessing step took 14 hours. All experiments are run on Goolge ColabPro+, which has the advantage of background execution and premium GPU support comparing to its free version. The TPC, transformer and traditional LSTM models typically require up to 3GB of CPU RAM and up to 3GB of GPU, while cw-LSTM model requires up to 30GB of GPU. All experiments combined consumed about 400+ compute units on ColabPro+, of which the two CW-LSTM models used 100 compute units each. All experiments are run on Tesla T4 GPU (standard GPU acceleration in ColabPro+), with the exception of the two CW-LSTM which are run on NAVIDA A100 GPU (premium GPU acceleration in ColabPro+) due to it's much heavier consumption of computing resource. A summary of computing times of different models are listed in Table 2.

4 Results

Overall, we are able to replicate the main findings in the original paper and demonstrate the superior performance of TPC model in the task of ICU LOS prediction.

Table 2: Computational times for different models

Model	Hours
TPC	3
LSTM	0.75
CW-LSTM	4.8
Transformer	2.2
Point. Only	1.8
Temp. Only	2
TPC(no skip)	2
TPC (no diag)	3
TPC(MSE)	3
Transformer(MSE)	2.2
LSTM(MSE)	0.75
CW-LSTM(MSE)	4.8

4.1 Model performance when trained with MSLE vs MSE

Results are listed in Table 3. Consistent with findings from the paper, we can see that using the MSLE (rather than MSE) loss function leads to significant improvements in all models, with large performance gains in MAD, MAPE, MSLE and Kappa,

4.2 eICU data preprocessing pipeline analysis

The data processing pipeline developed for eICU database in the paper involves several steps:

- **Data Cleaning:** A series of data cleaning methods such as imputing missing values, removing outliers, and normalizing the data are used to resolve the missing values, outliers, and inconsistencies issues in raw data.
- **Feature Engineering:** This step extracts static and temporal features from the data, such as demographics, vital signs, and lab test results. And also create new features, such as the difference between consecutive measurements, to capture the temporal dynamics of the data.
- **Temporal Alignment:** This step align the data in a consistent manner using fixed-length non-overlapping windows (e.g., 4-hour windows). This allows the model to analyze the data within a consistent time frame and capture temporal patterns.
- **Data Transformation:** This step transforms the data to make it suitable for the TPCN

Table 3: Comparision when using MSLE vs MSE

Loss function	Model	MAD	MAPE	MSE	MSLE	R^2	Kappa
MSLE	TPC	1.74	41.86	22.02	0.47	0.37	0.67
	CW-LSTM	2.68	123.54	32.25	1.49	0.08	0.30
	LSTM	2.67	133.17	31.46	1.54	0.10	0.32
	Transformer	2.65	117.45	32.19	1.47	0.08	0.30
MSE	TPC	2.21	111.13	22.13	1.90	0.37	0.55
	CW-LSTM	2.83	211.11	30.72	1.87	0.12	0.28
	LSTM	2.93	249.57	30.73	2.06	0.12	0.25
	Transformer	2.89	241.17	30.36	2.00	0.13	0.26

model. It uses one-hot encoding for categorical variables and normalize the continuous variables to have zero mean and unit variance.

- **Data Splitting:** The dataset is split into training, validation, and test sets. This allows us to train the model on one set of data, tune the hyperparameters using the validation set, and evaluate the model's performance on the test set.

Upon analyzing the eICU data pipeline, we have identified its potential in handling sparsity and missing data as well as its adaptability to other EHR databases. To further assess these capabilities, we plan to delve deeper and investigate whether the approach is indeed generalizable to other EHR databases. This comprehensive analysis will provide us with a more robust understanding of the eICU data pipeline's effectiveness and generalizability.

4.3 Further plans

We plan to explore whether certain differences in model architecture would impact its performance. In particular, we plan to compare the performance of the TPC model, temporal-only model, pointwise-only model, temporal-only model with weight sharing, and TPC model without skip connections. Such analysis would be interesting as it would demonstrate the true key components leading to performance improvement in the TPC model, while also shed light on the model's interpretability.

5 Communication with original authors

We submitted tickets in github repo (https://github.com/EmmaRocheteau/TPC-LoS-prediction/issues/10).

References

Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Emma Rocheteau, Pietro Liò, and Stephanie Hyland. 2021. Temporal pointwise convolutional networks for length of stay prediction in the intensive care unit. In *Proceedings of the conference on health, inference, and learning*, pages 58–68.

Seyedmostafa Sheikhalishahi, Vevake Balaraman, and Venet Osmani. 2019. Benchmarking machine learning models on eicu critical care dataset. *arXiv preprint arXiv:1910.00964*.

Huan Song, Deepta Rajan, Jayaraman Thiagarajan, and Andreas Spanias. 2018. Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.