Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit Paper Reproduction Study

Cheng Wei and Kinara Pandya
{chengw5, kinarap2}@illinois.edu

Group ID: 02, Paper ID: 188, Difficulty: Hard
Presentation link: https://youtu.be/-TeB2xq7Dbk
Code link: https://github.com/weicheng113/DL4H-Project

1 Introduction

In this project we aim to replicate the paper "Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit" by Emma Rocheteau, Pietro Liò and Stephanie Hyland [1]. The original paper proposes a new model architecture named Temporal Pointwise Convolution (TPC) that combines temporal convolution and pointwise convolution, to predict the remaining length of stay of ICU patients in the eICU and MIMIC-IV datasets. This can help hospitals efficiently manage ICU beds and schedule their staff accordingly.

The authors claim that TPC mitigates many common challenges with Electronic Health Records, such as skewness, irregular sampling and missing data which results in significant improvement over traditional timeseries models like LSTM and Transformer.

2 Scope of reproducibility

The TPC model will achieve higher accuracy on a test dataset than LSTM and transformer models when all of them are well trained on the eICU dataset.

2.1 Addressed claims from the original paper

The following claims will be tested:

- TPC has anywhere from 18% to 68% performance gain over the commonly used Long-Short Term Memory (LSTM) network, and the multi-head self-attention network known as the Transformer.
- Using mean-squared logarithmic error (MSLE) as the loss function to train length of stay models vastly increases the accuracy of the models as it deals more naturally with positively-skewed labels.
- Adding in-hospital mortality as a side-task in the TPC model will further increase the accuracy.

3 Methodology

3.1 Model descriptions

The proposed TPC model consists of two major components, which are temporal convolution and pointwise convolution. The two components are combined to exploit both temporal patterns of a feature across time and relationships among different features within the measurements. The TPC model contains about 1.088M parameters. The goal of TPC model is to minimize the difference between the predicted length of stay and the actual length of stay. In this context, the loss function of mean sqared log error (MSLE) is more reasonable choice than mean squared error (MSE). We would like to penalize proportional errors, as there is a difference for a 2 days prediction error in 4-day stay and 20-day stay. This is also confirmed by the better resulting model performance with MSLE than with MSE in the result section. Additionally, there is a side mortality task introduced with binary cross entropy loss to measure to discrepancy between true distribution and predicted distribution. With the combined weighted loss, the performance of the model has been further improved.

Let us examine the overall concepts of the two major components of TPC separately as shown in Figure 1. In part (a), the temporal convolution layers are displayed from bottom(layer 1 with dilation d=1) to top(layer 3 with dilation d=3). 1-D convolution with some kernel size and filter group size is applied to the input features to extract temporal patterns across a single feature. In order to utilize further information from the past, dilation of convolution is used such that at layer 3 it covers a larger range from the past. A left padding

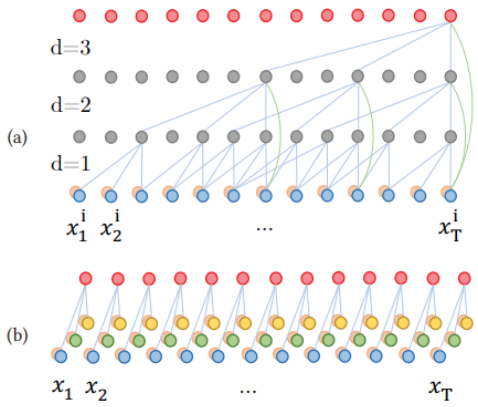

Figure 1: (a) Temporal convolution (b) Pointwise convolution.

is applied to the input features so that the future information is not leaked into the current step. In part (b), pointwise convolution or 1x1 convolution is used to extract the information across different features at the current time point.

This figure shows a layer of the TPC model:

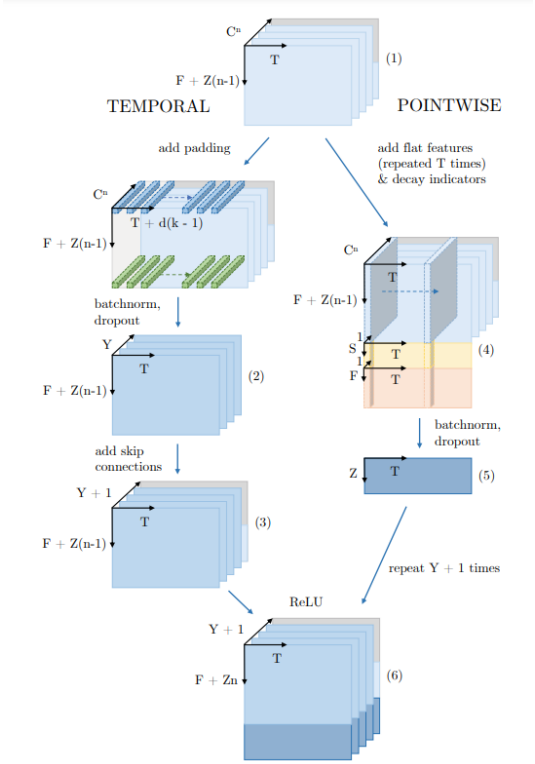

Figure 2: A layer of the TPC model.

Next, let us look at the TPC model architecture in detail. Figure 2 shows one layer of the TPC model - at n_{th} layer, for a single patient. (1) Firstly, the input of a patient to the n_{th} layer is of shape $T \times (F + Z_{n-1}) \times C^n$, where T is number of measurements across time, F is original features from measurements, Z_{n-1} is pointwise feature from previous $n-1$ layer and C^n is channel size of a feature. Namely a patient has T measurements across time, each measurement consists of $F + Z_{n-1}$ features, and each feature is described by C^n channels. The channels of a feature are feature value and decay indicator pair at 1^{st} layer. From 2^{nd} layer onwards, the channels of a feature is either from concatenation of convolution filter output and skip connection of original feature F or repeated values of pointwise feature Z_{n-1}.

The left part of diagram represents the flow of a temporal convolution component. A $C^n \times k$ (k is kernel size) filter group convolves along time axis T to exact single feature patterns across time (which is followed by batch normalization and dropout). (2) The output of convolution is of same shape except that it now has Y channels from different filters. (3) Then an original feature value or previous pointwise value is appended to the output by skip connection so that the channel size is now $Y + 1$. The right part of the diagram shows the process of pointwise convolution. Features from the input (1)(output from previous layer), pointwise convolution result of the previous layer, original features and static features are concatenated together. (4) Multiple pointwise convolution filters are applied on the concatenated features at the current time point to extract interaction information among features(pointwise convolution, also referred as 1x1 convolution, is equivalent to a linear layer). (5) After batch normalization and dropout, the output is of shape $T \times Z$, where T is number of measurements and Z is the number of extracted pointwise features. At the bottom of the diagram, (6) the outputs from both temporal convolution and pointwise convolution(the value is repeated $Y + 1$ times to match the shape of temporal convolution output) are concatenated together along the feature dimension and a ReLU activation function is applied.

Overall, the TPC model consists of a stack of TPC layers, followed by fully connected layers and activation functions.

The table below lists the parameters used by the TPC model and their default values:

Parameter	Default value
Number of layers	9
Kernel size	4
No. temporal kernels	12
Point size	13
Share weights	False
No skip connections	False

Table 1: Default parameter values for the TPC model.

3.2 Data descriptions

We used the eICU Collaborative Research Database[2] available through PhysioNet[3] which comprises of 200,859 patient unit encounters for 139,367 unique patients admitted to ICUs between 2014 and 2015. The following preprocessing steps were taken prior to using this data for model training:

- Adult patients ($\geq$18 years) with an ICU length of stay of at least 5 hours and at least one recorded observation were selected.
- 87 time series features were selected from the following tables: lab, nursecharting, respiratorycharting, vitalperiodic and vitalaperiodic.
- For a variable to be selected it had to be present in at least 12.5 % of patient stays, or 25% for lab variables.
- Missing data in time series data was forward-filled over the gaps.
- 'decay indicators' were added to specify location of stale data.The decay was calculated using the formula 0.75^j , where j is the time since the last recording .
- Diagnoses data was extracted from the pasthistory, admissiondx and diagnoses table and 17 static features from the patient, apachepatientresult and hospital tables.

The patients were divided as follows:

- Training set - 70%
- Validation set - 15%
- Testing set - 15%

The table below shows the cohort summary for the eICU dataset:

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)	2 54.1%
Age (mean)	63.1
LoS (mean)	3.01
LoS (median)	1.82
Remaining LoS (mean)	3.47
Remaining LoS (median)	1.67
In-hospital mortality	9.25%
Number of input features	104
Time series	87
Static	17

Table 2: Cohort summary for the eICU dataset.

3.3 Hyperparameters

We used the best hyperparameters given by the author in the source code except for the batchsize as it is restricted by the configuration of the machine/s used to train the model. Please refer to the hyperparameters table for more information.

Model	Num Layers	Batchsize	Epochs
CW LSTM	2	160	30
Transformer	6	64	15
TPC	9	64	15
Point. only	9	512	15
Temp. only	9	64	15
TPC(multitask)	9	24	15
TPC(no skip)	9	64	15
TPC(MSE)	9	24	15

Table 3: Hyperparameters

3.4 Implementation

For the paper reproduction, we chose to re-use most of the author's code and re-implement some of the parts to make it run efficiently without any errors. We re-implemented the data loading part with a multi-process data loader so that we can fully utilize GPU resources, which reduced the training time of a TPC model from more than 16 hours to around 10 hours. Additionally, we re-implemented

the data shuffling process with random data item access support, which further reduced the model training time for a TPC model from about 10 hours to approximately 4 hours. For additional experiments, we modified the model source code to include decay indicator also into the skip connection.

Code link: https://github.com/weicheng113/DL4H-Project

3.5 Computational requirements

It took about 4 hours to create the eICU database in PostgreSQL and required 60GB disk space to store it and an additional 8 hours to execute the preprocessing scripts. The size of the processed dataset is 21GB. All model experiments were done on AWS EC2 g4dn.2xlarge NVIDIA T4 GPU 16GB, 8 vCPUs and 32 GB memory for this project. The total EC2 g4dn.2xlarge usage is 92.696 hours, part of which was spent on model resource utilization fine-tuning at initial stage. The final training time for different models are recorded in the table below.

Model	Hours	Est. hours	Epochs
CW LSTM	13.2	20	30
Transformer	5.2	16	15
TPC	4.3	16	15
TPC(multitask)	4.4	16	15
TPC(no skip)	4.2	16	15
TPC(MSE)	4.5	16	15
Point. only	3.5	16	15
Temp. only	4.1	16	15

Table 4: Training time for models

4 Results

We trained the TPC model on the eICU dataset for 15 epochs and computed the evaluation metrics on the test dataset. It is comparable with the values in the original paper. We were able to get the evaluation metrics within 10 % of the author's values for all the experiments. In subsection 4.1 we compare TPC to the models that are usually used for timeseries data like LSTM and Transformer. We see approximately 60 % performance boost over the conventional models. In subsection 4.2, we compare TPC to other TPC model variants where TPC(multitask) performs the best.

Paper	MAD	MSE	MAPE
Original	1.78±0.02	21.7±0.5	63.5±4.3
Replication	1.658	19.539	49.697

Table 5: Result comparison for TPC model on eICU test data.

Paper	MSLE	R^2	KAPPA
Original	0.70±0.03	0.27±0.02	0.58±0.01
Replication	0.458	0.443	0.710

Table 6: Result comparison for TPC model on eICU test data.

4.1 TPC vs Baseline models

Model	MAD	MSE	MAPE
TPC	1.658	19.539	49.697
CW LSTM	2.631	31.825	115.745
Transformer	2.592	30.617	126.388

Table 7: Result comparison for TPC model vs other baseline models.

Model	MSLE	R^2	KAPPA
TPC	0.458	0.443	0.710
CW LSTM	1.424	0.096	0.324
Transformer	1.437	0.127	0.340

Table 8: Result comparison for TPC model vs other baseline models.

4.2 TPC vs TPC variants

Model	MAD	MSE	MAPE
TPC	1.658	19.539	49.697
TPC(multitask)	1.241	16.058	27.520
TPC(no skip)	1.918	22.929	64.399
TPC(MSE)	2.107	20.965	144.790
Point. only	2.665	31.458	110.025
Temp. only	1.798	21.437	57.617

Table 9: Result comparison for TPC model vs other TPC variant models.

Model	MSLE	R^2	KAPPA
TPC	0.458	0.443	0.710
TPC(multitask)	0.214	0.542	0.821
TPC(no skip)	0.716	0.346	0.627
TPC(MSE)	1.565	0.402	0.639
Point. only	1.517	0.106	0.336
Temp. only	0.615	0.389	0.665

Table 10: Result comparison for TPC model vs other TPC variant models.

4.3 Additional results not present in the original paper

We modified the existing model to include decay indicator into skip connection in addition to the feature value. As decay indicator is for indicating how fresh the corresponding feature value is, it is useful information to pair with its feature value. We did an experiment on TPC model and saw about 30 % improvement in MAD performance in the results below. (Note that adding decay indicator into skip connection also increases model size slightly).

Model	Model Size	MAD	MSE
TPC	1.088M	1.658	19.539
TPC(skip decay)	1.151M	1.267	15.385

Table 11: TPC vs TPC(skip decay).

Model	MAPE	MSLE	R^2	Kappa
TPC	49.697	0.458	0.443	0.710
TPC(skip decay)	30.231	0.222	0.561	0.817

Table 12: TPC vs TPC(skip decay).

5 Discussion

Due to time constraints, we did not run the all the experiments done in the original paper, which includes 10 runs for different models to calculate confidence intervals for performance, experiments involving the MIMIC-IV data set, some baselines and some of the ablations studies like Temporal only (WS), TPC (no diag.), TPC (no decay) and Point(no decay). The performance of the two baseline models were worse than the scores given in the paper(we contacted the author and have not found the reason yet). The values for the experiments we had done for the main TPC model and the various ablation studies, largely matched with what the author had claimed in the paper. Most of results of our experiments were slightly better than the ones in the paper, which could be due to randomness in the experiments and a new shuffling strategy introduced. Our experiment results also confirmed that the carefully crafted TPC model outperformed the two strong previous models, channel-wise LSTM and Transformer.

5.1 What was easy

- The source code repository by the author was complete and well organized. It included proper documentation and all the baseline models implementations.
- The author had a short youtube video explaining the paper and any questions we asked the author were promptly answered.

5.2 What was difficult

- We picked a hard paper to reproduce for the final project of our course. The TPC model proposed by the author is not a simple model made of combination of existing PyTorch modules, but it is a custom and complex model created from scratch. It includes a number of new ideas and concepts that we had not learnt from our course like temporal convolution, dilation, filter groups and pointwise convolution. To understand the paper well, we had to do a thorough research on the relevant topics but there were very limited resources online. Eventually, we created some experimental examples ourselves to properly comprehend the relevant concepts.
- eICU data set is large. It took about 60 GB disk space to set up the database locally and took about 8 hours to pre-process the data from database to produce 21 GB of data, which included the train, validation and test data sets.
- It took a long time to train the models with the large data set. The TPC model took more than 16 hours with the default hyperparameters to train with data shuffling. With a couple of optimizations, we were able to reduce it to about 4 hours.

5.3 Recommendations for reproducibility

- The steps given by the author to set up the database locally did not work. We followed

different steps to create the database and documented them in our repositories README file.

- Using Conda environment yaml file is recommended (which the original source code does not contain) for creating a new Conda environment with the desired Python version.

6 Communication with original authors

- An GitHub issue was created to discuss project setup problem. There was a batch bug fixed by us in order to train the model.
- An GitHub issue was created to confirm conceptual understanding of the TPC model.
- An GitHub issue was created to discuss some resulting metric mismatch with the paper.

References

[1] Emma Rocheteau, Pietro Liò, and Stephanie Hyland. 2021. Temporal Pointwise Convolutional Networks for Length of Stay Prediction in the Intensive Care Unit. In ACM Conference on Health, Inference, and Learning (ACM CHIL '21), April 8–10, 2021, Virtual Event, USA. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3450439.3451860

[2] Tom J Pollard, Alistair E W Johnson, Jesse D Raffa, Leo A Celi, Roger G Mark, and Omar Badawi. 2018. The eICU Collaborative Research Database, A Freely Available Multi-Center Database for Critical Care Research. Scientific Data 5, 1 (2018), 180178.

[3] A. L. Goldberger, L. A. N. Amaral, L. Glass, et al. 2000. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. Circulation 101, 23 (2000), e215–e220