Preprint 2020. 1–26

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

Emma Rocheteau ECR38@CAM.AC.UK
University of Cambridge

Pietro Liò PL219@CAM.AC.UK
University of Cambridge

Stephanie Hyland STEPHANIE.HYLAND@MICROSOFT.COM
Microsoft Research

Abstract

The pressure of ever-increasing patient demand and budget restrictions make hospital bed management a daily challenge for clinical staff. Most critical is the efficient allocation of resource-heavy Intensive Care Unit (ICU) beds to the patients who need life support. Central to solving this problem is knowing for how long the current set of ICU patients are likely to stay in the unit. In this work, we propose a new deep learning model based on the combination of temporal convolution and pointwise (1x1) convolution, to solve the length of stay prediction task on the eICU critical care dataset. The model – which we refer to as Temporal Pointwise Convolution (TPC) – is specifically designed to mitigate for common challenges with Electronic Health Records, such as skewness, irregular sampling and missing data. In doing so, we have achieved significant performance benefits of 18-51% (metric dependent) over the commonly used Long-Short Term Memory (LSTM) network, and the multi-head self-attention network known as the Transformer.

1. Introduction

In-patient length of stay (LoS) explains approximately 85-90% of inter-patient variation in hospital costs in the United States (Rapoport et al., 2003). Extended length of stay is associated with increased risk of contracting hospital acquired infections (Hassan et al., 2010) and mortality (Laupland et al., 2006). Hospital bed planning can help to mitigate these risks and improve patient experiences (Blom et al., 2015). This is particularly important in the intensive care unit (ICU), which has the highest operational costs in the hospital (Dahl et al., 2012) and a limited supply of specialist staff and resources.

At present, discharge date estimates are done manually by clinicians, but these rapidly become out-of-date and can be unreliable (Nassar and Caruso, 2016)[1]. Automated systems drawing on the electronic health record (EHR) may reduce the administrative burden on clinicians and enable sophisticated planning strategies e.g. scheduling high-risk elective surgeries on days with more availability predicted in the ICU (Gentimis et al., 2017).

In our work, we predict the patients' remaining ICU length of stay at hourly intervals during their stay using preceding data from the EHR (similar to Harutyunyan et al. (2019)). When designing both the architecture and pre-processing, we focus on mitigating for the effects of non-random missingness due to irregular sampling, sparsity, outliers,

1. Mak et al. (2012) found that the average error made by clinicians when was 3.82 days.

skew, and other common biases in EHR data. Our key contributions are:

1. A new model – Temporal Pointwise Convolution (TPC) – which combines: 1) Temporal convolutional layers (van den Oord et al., 2016; Kalchbrenner et al., 2016), which capture causal dependencies across the time domain. 2) Pointwise convolutional layers (Lin et al., 2013), which compute higher level features from interactions in the feature domain. We show that these methods complement each other by extracting different information. Our model outperforms the commonly used Long-Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) and the Transformer (Vaswani et al., 2017).

2. We make a case for using the mean squared logarithmic error (MSLE) loss function to train LoS models, as it helps to mitigate for positive skew.

3. We present a visualisation of the model's performance to show the reliability as a function of the time since admission and the predicted remaining LoS.

Our code is available at: https://github.com/EmmaRocheteau/eICU-LoS-prediction.

2. Related Work

Despite its importance, the LoS prediction task has received less attention than mortality prediction. This could be on account of its difficulty; LoS depends on operational factors and there is considerable positive skew in the distribution (see Figure 1). While LoS has been addressed as a regression problem, optimised using the mean-squared error (MSE)(Purushotham et al., 2018; Sheikhalishahi et al., 2019), it is often simplified into binary classification (short vs. long stay)(Gong et al., 2017; Nestor et al., 2018; Rajkomar et al., 2018), or as a multi-class task (Harutyunyan et al., 2019). This simplification comes at a cost of utility, so we focus on the (harder) regression variant.

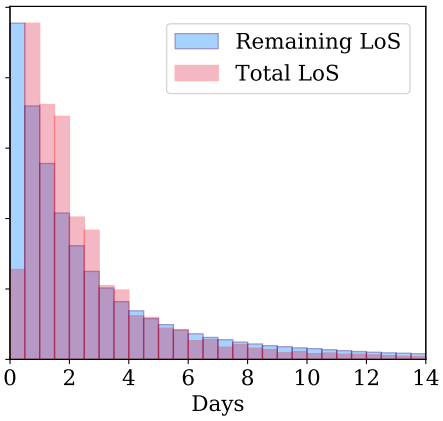

Figure 1: Total and remaining length of stay distributions. The mean and median values are (2.99, 1.82) and (3.47, 1.67) days respectively.

Owing to the centrality of time series in the EHR, LSTMs have been by far the most popular model for predicting LoS and have achieved state-of-the-art results (Harutyunyan et al., 2019; Sheikhalishahi et al., 2019; Rajkomar et al., 2018). This reflects the prominence of LSTMs in other clinical prediction tasks such as predicting in-hospital adverse events including cardiac arrest (Tonekaboni et al., 2018) and acute kidney injury (Tomašev et al., 2019), forecasting diagnoses, medications and interventions (Choi et al., 2015; Lipton et al., 2015; Suresh et al., 2017), missing-data imputation (Cao et al., 2018), and mortality prediction (Che et al., 2018; Harutyunyan et al., 2019; Shickel et al., 2019). More recently, the Transformer model (Vaswani et al., 2017), has been shown to marginally outperform the LSTM on LoS (Song et al., 2018). Therefore, the LSTM and the Transformer were chosen as key baselines.

3. Methods

Model Overview We want our model to extract both temporal trends and inter-feature relationships to capture the patient's clinical state and make accurate LoS predictions. Consider a patient who is experiencing slowly worsening respiratory symptoms but is otherwise stable. As this patient is unlikely to be weaned from their ventilator in the near future, a clinician would likely anticipate a long remaining LoS, but how do they come to this conclusion? Intuitively, they are evaluating the trajectory of the patient, so they may ask themselves "is the respiratory rate getting better or deteriorating?". They might also want to obtain a better indication of lung function by combining certain features e.g. the PaO_2/FiO_2 ratio. In short, they focus on trends and feature interactions – both of which we aim to extract from our model.

Formally, our task is to predict the remaining LoS at regular timepoints $y_1, \ldots, y_T \in \mathbb{R}_{>0}$ up to the discharge time T, using the diagnoses ($\mathbf{d} \in \mathbb{R}^{D\times 1}$), static features ($\mathbf{s} \in \mathbb{R}^{S\times 1}$), and time series ($\mathbf{x}_1, \ldots, \mathbf{x}_T \in \mathbb{R}^{F\times 2}$). Initially, for every timepoint t, there are two 'channels' per time series feature: F feature *values*, $\mathbf{x'}_t \in \mathbb{R}^{F\times 1}$, and their corresponding decay indicators[2], $\mathbf{x''}_t \in \mathbb{R}^{F\times 1}$. Next, we explain how we extract temporal trends and inter-feature relationships using a novel combination of techniques.

Temporal Convolution Temporal Convolution Networks (TCNs) (van den Oord et al., 2016; Kalchbrenner et al., 2016) are a subclass of convolutional neural networks (Fukushima, 1980) that convolve over the time dimension. They have two key principles: the output is the same length as the input, and there can be no leakage of data from the future. We use stacked TCNs to extract *temporal trends* in our data. Unlike most implementations, we *do not share weights across features* i.e. weight sharing is only across time (like in Xception (Chollet, 2016)). This is because our features differ sufficiently in their temporal characteristics to warrant specialised processing.

We define the temporal convolution operation for the i_{th} feature in the n_{th} layer as

$$(f^{(n,i)} * \mathbf{h}^{(n,i)})(t) = \sum_{j=1}^{k} f^{(n,i)}(j)\mathbf{h}^{(n,i)}_{t-d(j-1)} \quad (1)$$

where $\mathbf{h}^{(n,i)}_{1:t} \in \mathbb{R}^{C^{(n)}\times t}$ represents the temporal input to layer n up to timepoint t, which contains $C^{(n)}$ channels per feature[3]. The convolutional filter $f^{(n,i)} : \{1, \ldots, k\} \to \mathbb{R}^{Y\times C^{(n)}}$ is a tensor of $Y \times C^{(n)} \times k$ parameters per feature. It maps $C^{(n)}$ input channels into Y output channels while examining k timesteps. The output is therefore $(f^{(n,i)} * \mathbf{h}^{(n,i)})(t)^\top \in \mathbb{R}^{1\times Y}$. The dilation factor, d, and kernel size, k, together determine the temporal receptive field or 'timespan' of the filter: $d(k-1)+1$ hours for a single layer. The $t - d(j-1)$ term ensures that we only look backwards in time[4]. The receptive field can be increased by stacking multiple TCNs (as in Wavenet (van den Oord et al., 2016) and ByteNet (Kalchbrenner et al., 2016)). We increment the dilation by 1 with each layer i.e. $d = n$.

We concatenate the temporal convolution outputs for each feature, i as follows

$$\underbrace{(f^{(n)} * \overbrace{\mathbf{h}^{(n)}}^{\text{Temp. In.}(1)})}_{\text{Temp. Out.}(2)}(t) = \Big\|_{i=1}^{R^{(n)}} (f^{(n,i)} * \mathbf{h}^{(n,i)})(t)^\top \quad (2)$$

We use $\|$ to denote concatenation i.e. $\|_{i=1}^{A} \mathbf{a}^{(i)} = \mathbf{a}^{(1)} \,\|\, \ldots \,\|\, \mathbf{a}^{(A)}$. In our case, the output dimensions are $R^{(n)} \times Y$, where $R^{(n)}$ is the number of temporal input features.

2. The decay indicators tell the model how recently the observation $\mathbf{x'}_t$ was recorded. They are described in detail in Section 4.

3. In the first layer, the input $\mathbf{h}^{(n,i)}_{1:t}$ is the original data $\mathbf{x}^{(n,i)}_{1:t} \in \mathbb{R}^{2\times t}$, so $C^{(1)} = 2$.

4. To ensure that the output is always length T, we add left-sided padding of size $d(k-1)$ before every temporal convolution (not shown in equation 1).

Throughout this section we label terms with numbers (1), (2) etc. corresponding to objects in Figure 3. We recommend following this alongside the equations.

Pointwise Convolution Pointwise convolution (Lin et al., 2013), also referred to as 1×1 convolution, is typically used to reduce the channel dimension when processing images (Szegedy et al., 2014). It can be conceptualised as a fully connected layer, applied separately to each timepoint (shown diagrammatically in Figure 2). As in temporal convolution, the weights are shared across all timepoints; however, there is no *information transfer* across time. Instead, information is shared across the *features* to obtain Z interaction features[5], $\mathbf{p}_t^{(n)} = (\flat(\mathbf{h}_t^{(n)}) \parallel \mathbf{s} \parallel \mathbf{x''}_t) \in \mathbb{R}^{P^{(n)} \times 1}$, where $P^{(n)} = (R^{(n)} \times C^{(n)}) + F + S$, and $\flat : A^{d_1 \times d_2 \ldots \times d_n} \rightarrow A^{(d_1 \cdot d_2 \ldots \cdot d_n) \times 1}$ is the flatten operation. We define the pointwise convolution operation in the n_{th} layer as

$$\underbrace{(g^{(n)} * \overbrace{\mathbf{p}^{(n)}}^{\text{Point. In.(4)}})(t)}_{\text{Point. Out.(5)}} = \sum_{i=1}^{P^{(n)}} g^{(n)}(i) p_t^{(n,i)} \tag{3}$$

where $g^{(n)} : \{1, \ldots, P^{(n)}\} \rightarrow \mathbb{R}^{Z \times 1}$ is the pointwise filter, and the resulting convolution produces Z output channels, so $(g^{(n)} * \mathbf{p}^{(n)})(t) \in \mathbb{R}^{Z \times 1}$.

Skip Connections We propagate skip connections (He et al., 2015) to allow each layer to see the original data and the pointwise outputs from previous layers. This helps the network to cope with sparsely sampled data. For example, suppose a particular blood test is taken once per day. In order to not to lose temporal resolution, we forward-fill these data (Section 4) and convolve with increasingly dilated temporal filters until we find the appropriate width to capture a useful trend. However, if the smaller filters in previous layers (which did not see any useful trend) have polluted the original data by reweighting, learning will be harder. Skip connections provide a consistent anchor to the input. They are concatenated (like in DenseNet (Huang et al., 2017), and are arranged in the shared-source connection formation (Wang et al., 2018)) as illustrated in Figure 2. The skip connections expand the feature dimension, $R^{(n)} = F + Z(n-1)$, to accommodate the pointwise outputs, and also the channel dimension to fit the original data, $C^{(n)} = Y + 1$. This is best visualised in Figure 3.

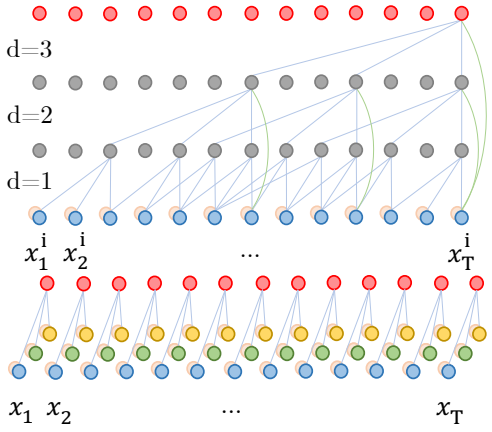

Figure 2: Top: Temporal convolution with skip connections. Each time series, i (blue) and their decay indicators (orange) are processed with independent parameters. Bottom: Pointwise convolution. There is no information sharing across time, only across features (blue, green, yellow).

Temporal Pointwise Convolution Our model – which we refer to as Temporal Pointwise Convolution (TPC) – combines temporal and pointwise convolution in parallel. Firstly, the temporal output is combined with the skip connections to form $\mathbf{r}_t^{(n)}$ (Step 3 in Figure 3).

$$\underbrace{\mathbf{r}_t^{(n)}}_{(3)} = \underbrace{(f^{(n)} * \mathbf{h}_t^{(n)})}_{\text{Temp. Out.(2)}} \parallel \underbrace{\mathbf{x'}_t \parallel \left[\Big\Vert_{n'=1}^{n-1} (g^{(n')} * \mathbf{p}_t^{(n')}) \right]}_{\text{Skip Connections}} \tag{4}$$

5. We use a wider set of features for pointwise convolution, including static features $\mathbf{s}$ and decay indicators $\mathbf{x''}$.

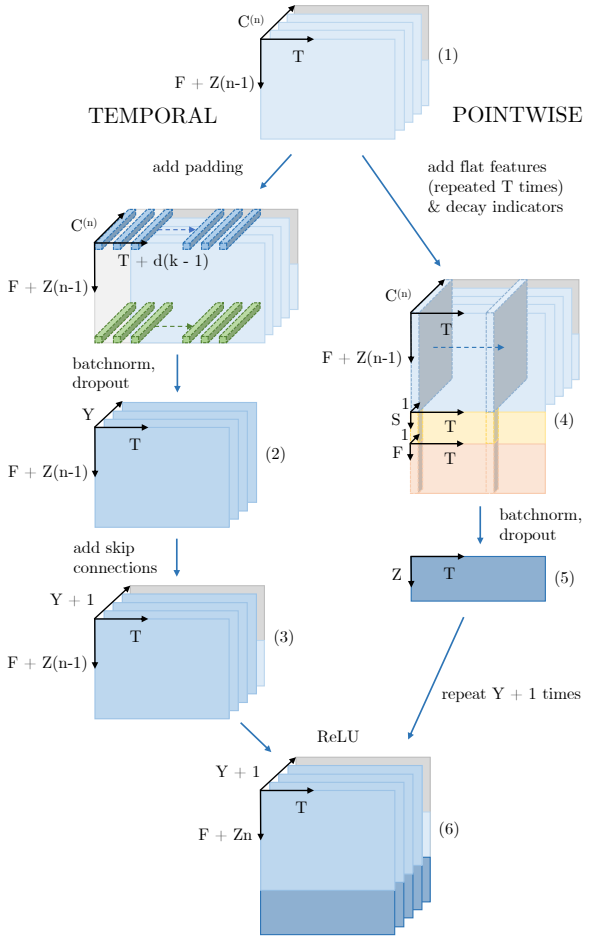

Figure 3: The n_{th} TPC layer. Left-sided padding (white) is added to the temporal side before each feature is processed independently. On the pointwise side, flat features (yellow) and decay indicators (orange) are added before each convolution.

$\mathbf{r}_t^{(n)}$ is then concatenated with the pointwise output after it has been repeated $Y+1$ times. We can therefore define the n_{th} TPC layer as

$$\underbrace{\mathbf{h}_t^{(n+1)}}_{\text{TPC Out.(6)}} = \sigma\Bigg(\underbrace{\mathbf{r}_t^{(n)}}_{(3)} \,\|\, \Bigg[\,\Big\|_{i=1}^{Y+1} \underbrace{(g^{(n)} * \mathbf{p}_t^{(n)})}_{\text{Point. Out.(5)}}\Bigg]\Bigg) \quad (5)$$

where σ represents the ReLU activation function. The full model has N TPC layers stacked sequentially (9 in our model). After N layers, the output $\mathbf{h}_t^{(N)}$ is combined with static features $\mathbf{s} \in \mathbb{R}^{S\times 1}$, and a diagnosis embedding $\mathbf{d}^* \in \mathbb{R}^{D^*\times 1}$. Two pointwise layers are then applied to obtain the final predictions (see Appendix A for the full details). We use batch normalisation (Ioffe and Szegedy, 2015) and dropout (Srivastava et al., 2014) throughout to regularise the model.

Loss Function The remaining LoS has a positive skew (shown in Figure 1) which makes the prediction task more challenging. We address this by replacing the commonly-used mean squared error (MSE) loss with mean squared *log* error (MSLE).

$$\mathcal{L} = \frac{1}{T}\sum_{t=1}^{T}(\log(\hat{y}_t) - \log(y_t))^2 \quad (6)$$

MSLE penalises *proportional* errors, which is more reasonable when considering an error of e.g. 5 days in the context of a 2-day stay vs. a 30-day stay. The difference can be seen in Figure 9 in Appendix E.4. For bed management purposes it is particularly important not to harshly penalise over-predictions. The model will become overly cautious and regress its predictions towards the mean. This is counterproductive because long stay patients have a disproportionate effect on bed occupancy.

4. Data

eICU Database We use the eICU Collaborative Research Database (Pollard et al., 2018), a multi-centre dataset collated from 208 care centres in the United States. It comprises 200,859 patient unit encounters for 139,367 unique patients admitted to ICUs between 2014 and 2015.

Remaining Length of Stay Task We assign a remaining LoS target to each hour of the stay, beginning at 5 hours and ending when the patient dies or is discharged. We only train on data within the first 14 days of any patient's stay to protect against very long batches which would slow down training. This cut-off applies to 2.4% of patient

stays, but it does *not* affect their maximum remaining length of stay values.

Cohort and Feature Selection We selected all adult patients (>18 years) with an ICU LoS of at least 5 hours and at least one recorded observation, resulting in 118,534 unique patients and 146,666 ICU stays. We selected time series from the following tables: *lab*, *nursecharting*, *respiratorycharting*, *vitalperiodic* and *vitalaperiodic*. To be included, variables had to be present in at least 12.5% of patient stays, or 25% for *lab* variables. As shown in Figure 4, the *lab* variables tend to be sparsely sampled. To help the model cope with this missing data, we forward-filled over the gaps[6] and added 'decay indicators' to specify where the data is stale. The decay was calculated as 0.75^j, where j is the time since the last recording. This is similar in spirit to the masking used by Che et al. (2018).

We extracted diagnoses from the *pasthistory*, *admissiondx* and *diagnoses* tables, and 17 static features from the *patient*, *apachepatientresult* and *hospital* tables (see Tables 2 and 3 in Appendix B for the full list of features and further details).

After pre-processing, the data were divided such that 70% of patients were used for training, 15% for validation, and 15% for testing.

5. Experiments and Results

Baselines We include 'mean' and 'median' models which always predict 3.5 and 1.7 days respectively (the mean and median of the training data). APACHE IV (Zimmerman et al., 2006) is a severity scoring model which is evaluated only once per patient at 24 hours. It cannot be compared directly, but we include it *only as a point of reference* for a widely used clinical model. We include a standard LSTM baseline which is very similar to Sheikhalishahi et al. (2019). We also include a channel-wise LSTM (CW LSTM) similar to Harutyunyan et al. (2019) which consists of a set of independent LSTMs that process each feature separately (note the similarity with the TPC model). Our Transformer baseline (Vaswani et al., 2017) is a multi-head self-attention model. Like the TPC, it is not constrained to progress one timestep at a time; however, unlike TPC, it is not able to scale its receptive fields or process features independently. Further implementation details including hyperparameters are in Appendix C.

6. This is preferable to interpolation as the clinician would only have the most recent value.

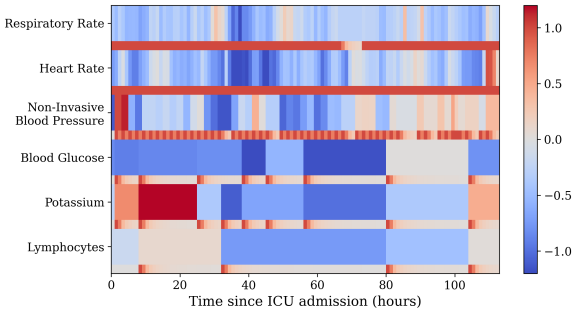

Figure 4: Example data from a patient (after pre-processing). The colour scale indicates the value of the feature, and the narrow bars show the corresponding decay indicators. Blood glucose, potassium and lymphocytes are from the *lab* table and are sparsely sampled. Non-invasive blood pressure is manually recorded by the nurse every 2 hours, while respiratory rate and heart rate are vital signs that are automatically logged.

Evaluation Metrics We report on 6 metrics for each experiment: mean absolute deviation (MAD), mean absolute percentage error (MAPE), mean squared error (MSE), mean squared log error (MSLE), coefficient of determination (R^2) and Cohen Kappa Score. This is important because bad models can 'cheat' particular metrics by being close to the mean or median value. See Appendix D for additional discussion of metrics.

TPC Performance Table 1a shows that the TPC model has outperformed all of the baseline models on every metric – particularly

Table 1: Performance of the TPC model compared to baseline models (a) and various ablation studies (b) and (c). Unless otherwise specified, the loss function is MSLE. For the first four metrics, lower is better. The error margins are 95% confidence intervals (CIs) calculated over 10 runs. (a) shows the baseline comparisons. Note that mean, median and APACHE are deterministic models so they have no error bars. †Note that the APACHE results cannot be compared directly to the other models (explained in 'Baselines'). (b) compares the effect of the loss function on the TPC model. See Table 10 for the MSE results of LSTM, CW LSTM and Transformer. (c) shows various TPC ablation studies. WS refers to weight sharing. The best results are highlighted in blue. If the result is statistically significant on a t-test then it is indicated with stars (* = p < 0.05, ** = p < 0.001). Results that are not significantly different from the best result are highlighted in light blue. The TPC (MSLE) result has been repeated in each subtable for ease of comparison.

	Model	**MAD**	**MAPE**	**MSE**	**MSLE**	R^2	**Kappa**
(a)	Mean	3.21	395.7	29.5	2.87	0.00	0.00
	Median	2.76	184.4	32.6	2.15	-0.11	0.00
	APACHE†	2.54	182.1	16.6†	1.10	-0.01	0.20
	LSTM	2.39±0.00	118.2±1.1	26.9±0.1	1.47±0.01	0.09±0.00	0.28±0.00
	CW LSTM	2.37±0.00	114.5±0.4	26.6±0.1	1.43±0.00	0.10±0.00	0.30±0.00
	Transformer	2.36±0.00	114.1±0.6	26.7±0.1	1.43±0.00	0.09±0.00	0.30±0.00
	TPC	**1.78±0.02****	**63.5±4.3****	**21.7±0.5****	**0.70±0.03****	**0.27±0.02****	**0.58±0.01****
(b)	TPC (MSLE)	**1.78±0.02****	**63.5±4.3****	**21.7±0.5**	**0.70±0.03****	**0.27±0.02**	**0.58±0.01****
	TPC (MSE)	2.21±0.02	154.3±10.1	**21.6±0.2**	1.80±0.10	**0.27±0.01**	0.47±0.01
(c)	TPC	**1.78±0.02**	**63.5±3.8***	**21.8±0.5**	**0.71±0.03***	**0.26±0.02**	**0.58±0.01**
	Point. only	2.68±0.15	137.8±16.4	29.8±2.9	1.60±0.03	-0.01±0.10	0.38±0.01
	Temp. only	1.91±0.01	71.2±1.1	23.1±0.2	0.86±0.01	0.22±0.01	0.52±0.01
	Temp. only (WS)	2.34±0.01	116.0±1.2	26.5±0.2	1.40±0.01	0.10±0.01	0.31±0.00
	TPC (no skip)	1.93±0.01	73.9±1.9	23.0±0.2	0.89±0.01	0.22±0.01	0.51±0.01
	TPC (no diag.)	**1.77±0.02**	65.6±4.1	**21.5±0.5**	**0.71±0.03***	**0.27±0.02**	**0.59±0.01**
	TPC (no decay)	1.84±0.01	64.5±3.0	22.5±0.3	0.77±0.02	0.24±0.01	0.56±0.01
	Point. (no decay)	2.90±0.18	179.1±17.4	34.2±4.6	1.80±0.05	-0.16±0.16	0.33±0.00

on those more robust to skewness: MAPE, MSLE and Kappa. Discounting APACHE, the best performing *baselines* are the Transformer and the channel-wise LSTM (CW LSTM), whose results are almost indistinguishable. This is consistent with Harutyunyan et al. (2019) (for CW LSTM) and Song et al. (2018) (for Transformers), who found small improvements over standard LSTMs.

MSLE Loss Function Table 1b shows that using the MSLE loss function (v. MSE) leads to significant improvements in the TPC model, with large performance gains in MAD, MAPE, MSLE and Kappa, while conceding little in terms of MSE and R^2. MSE results for the other models are in Appendix Table 10; they show a similar pattern.

Ablation Studies To understand the impact of each design choice for the TPC model, we study performance under different ablations. In Table 1c we can see that the temporal-only model is superior to the pointwise-only model, but the TPC model outperforms them both. The temporal-only model performs much better than its weight-sharing variant, suggesting that having inde-

pendent parameters per feature is important for the LoS task. Removing the skip connections reduces performance by 5-25%.

We also tested the models with partial data. Perhaps surprisingly, we found that the exclusion of diagnoses does not seem to harm the model. This could be because the relevant diagnoses for predicting LoS e.g. Acute Respiratory Distress Syndrome (ARDS), are discernible from the time series data alone e.g. PaO2, FiO2, PEEP, Tidal Volume etc.

The decay indicators contribute a small (but statistically significant) benefit to the TPC model. Their contribution is more obvious in the pointwise-only model[7]. All of the metrics see improvements of 5-23%.

We also tested the models with only the laboratory tests (which are infrequently sampled) and all other variables (which tend to be regularly monitored). The results (shown in Appendix E.2) indicate that the TPC model is better able to exploit disparate EHR time series than the baselines. They also show that the advantage of the CW LSTM over the standard LSTM is only apparent when the model has to process different types of time series simultaneously. Note that we perform a feature importance post-hoc study in Appendix E.1.

Visualisation The errors made by our model are not uniform across all inputs, and aggregate measures mask this variability in model performance. We visualise MAPE (chosen for its interpretability) as a function of the time since admission and the *predicted* remaining LoS. As the user knows this information at the time of prediction, they can rapidly assess the reliability. Figure 5 shows an example for the TPC model. We can see that the model is generally reliable except for when it predicts a long LoS during the first 1-2 days of admission. Additional investigation revealed these to be under-predictions. We believe that this is an effect of skewness – short stay patients are far more frequent than long stay patients, so the model requires 1-2 days of data to predict a long LoS.

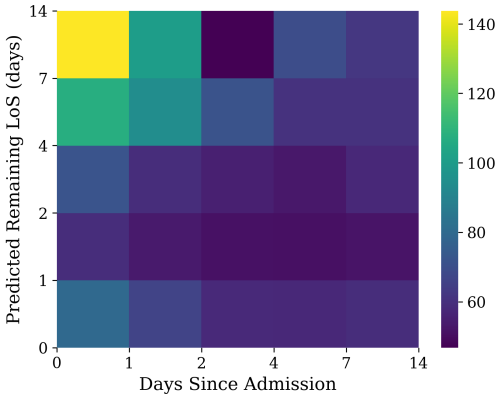

Figure 5: Mean absolute percentage error as a function of days since admission and predicted remaining LoS.

6. Discussion

We have shown that the TPC model outperforms all baseline models on the LoS task. To explain the success of TPC, we start by examining the parallel architectures in the TPC model. Each component has been designed to extract different information: trends from the temporal arm and inter-feature relationships from the pointwise arm. The ablation study reveals that the temporal element is more important, but their contributions are complementary as the best performance is achieved when they are used together.

Next, we highlight that the temporal-only model far outperforms its most direct comparison, the CW LSTM, on all metrics. Theoretically, they are well matched because they both have feature-specific parameters but are restricted from learning cross-feature interactions. To begin to explain this, consider how the information flows through the model. The temporal-only model can directly step across

7. They might reveal some of the temporal structure e.g. the pointwise model could learn links between up-to-date observations and patient deterioration.

large time gaps, whereas the CW LSTM is forced to progress one timestep at a time. This gives the CW LSTM the harder task of remembering information across a noisy EHR with distracting signals of varying frequency. In addition, the temporal-only model can tune its receptive fields for optimal processing of each feature thanks to the skip connections (which are not present in the CW LSTM).

The difference in performance between the temporal-only model with and without weight sharing provides strong evidence that assigning independent parameters to each feature is important. Some EHR time series are irregularly and sparsely sampled, and can exhibit considerable variability in the temporal frequencies within the underlying data (evident in Figure 4). This presents a challenge for any model, especially if it is constrained to learn one set of parameters to suit all features. The relative success of the CW LSTM over the standard LSTM when processing *disparate* time series – but not similar – also lends weight to this theory.

However, the assignment of independent parameters to each feature does not explain all the successes of TPC e.g. the TPC model can process disparate time series and gain more marginal performance than the CW LSTM. We need to consider that *periodicity* is a key property of EHR data[8]. The temporal component of the TPC model is the only architecture with an inherent periodic structure which makes it much easier to learn EHR trends. By comparison, a single attention head in the Transformer model does not look at timepoints a fixed distance apart, but can take an arbitrary form. This is a strength for natural language processing, given the variety of sentence structures possible, but it does not help the Transformer to process EHRs.

Finally, we reiterate that using MSLE loss instead of MSE greatly mitigates for positive skew in the LoS task, and this benefit is not model-specific (all of the baselines perform better with MSLE). This demonstrates that careful consideration of the task – as well as the data and model – is an important step towards building useful tools in healthcare.

Limitations and Future Work Our work has several important limitations. Firstly, prospective trials are needed to the prove real-world impact of sophisticated bed management systems (Rajkomar et al., 2018).

Secondly, we know that LoS is heavily influenced by operational factors, and clinical practices changing over time (Kalra et al., 2010). The eICU data has been collected over a short timespan from 2014 to 2015, so it would be instructive to test whether the methods are robust on the MIMIC-IV and AmsterdamUMCdb datasets in future work. Nevertheless, it is encouraging that our results – even hyperparameter values – are highly consistent with the work of Harutyunyan et al. (2019) and Song et al. (2018).

7. Conclusion

We have shown that the TPC model is well-equipped to analyse EHR time series containing missingness, differing frequencies and sparse sampling. We believe that the following four aspects contribute the most to its success:

1. The combination of two complementary architectures that are able to extract different features, both of which are important.
2. The ability to step over large time gaps.
3. The capacity to specialise processing to each feature (including the freedom to select the receptive field size for each).
4. The rigid spacing of the temporal filters, making it easy to derive trends.

8. This is true in sampling patterns and in the underlying biology e.g. medication schedules, sleep.

From a clinical perspective, we have contributed to the advancement of LoS prediction models, a prerequisite for automated bed management tools. This will have implications for cost reduction (Halpern and Pastores, 2015) and resource allocation (Mathews and Long, 2015) worldwide. From a computational perspective, we have provided key insights for retrospective EHR studies, particularly where LSTMs are the currently model of choice. In the broader context of machine learning for healthcare we have demonstrated that careful consideration of the complexities of health data is necessary to gain state-of-the-art performance in these tasks.

8. Acknowledgements

The authors would like to thank Alex Campbell, Petar Veličković, and Ari Ercole for helpful discussions and advice. We would also like to thank Louis-Pascal Xhonneux, Seyon Sivarajah, Rudolf Cardinal, Jacob Deasy, Paul Scherer, Ivan Beckley, and Katharina Kohler for their help in reviewing the manuscript. Finally we thank the Armstrong Fund, the Frank Edward Elmore Fund, and the School of Clinical Medicine at the University of Cambridge for their generous funding.

References

Mathias C. Blom, Karin Erwander, Lars M. Gustafsson, Mona Landin-Olsson, Fredrik Jonsson, and Kjell Ivarsson. The Probability of Readmission within 30 days of Hospital Discharge is Positively Associated with Inpatient Bed Occupancy at Discharge – A Retrospective Cohort Study. In *BMC Emergency Medicine*, 2015.

Andreas B Böhmer, Katja S Just, Rolf Lefering, Thomas Paffrath, Bertil Bouillon, Robin Joppich, Frank Wappler, and Mark U Gerbershagen. Factors influencing lengths of stay in the intensive care unit for surviving trauma patients: a retrospective analysis of 30,157 cases. *Critical care (London, England)*, 18(4):R143–R143, jul 2014.

Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. BRITS: Bidirectional Recurrent Imputation for Time Series. In *NeurIPS*, 2018.

Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan Liu. Recurrent Neural Networks for Multivariate Time Series with Missing Values. *Scientific Reports*, 8(1):6085, 2018.

Edward Choi, Mohammad Taha Bahadori, Andy Schuetz, Walter F. Stewart, and Jimeng Sun. Doctor AI: Predicting Clinical Events via Recurrent Neural Networks. *JMLR workshop and conference proceedings*, 56:301–318, 2015.

François Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. *CoRR*, abs/1610.02357, 2016.

Jacob Cohen. A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46, 1960.

Deborah Dahl, Greg G Wojtal, Michael Breslow, Randy Holl, Debra Huguez, David Stone, and Gloria Korpi. The High Cost of Low-Acuity ICU Outliers. *Journal of healthcare management / American College of Healthcare Executives*, 57:421–434, 2012.

Thuppahi Sisira De Silva, Don MacDonald, Grace Paterson, Khokan C. Sikdar, and Bonnie Cochrane. Systematized Nomenclature of Medicine Clinical Terms (SNOMED CT) to Represent Computed Tomography Procedures. *Comput. Methods Prog. Biomed.*, 101(3):324–329, 2011.

A Elixhauser, C Steiner, and L Palmer. Clinical Classifications Software, 2015.

Kunihiko Fukushima. Neocognitron: A Self-Organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4):193–202, 1980.

T. Gentimis, A. J. Alnaser, A. Durante, K. Cook, and R. Steele. Predicting Hospital Length of Stay Using Neural Networks on MIMIC III Data. In *2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing*, pages 1194–1201, 2017.

Jen J. Gong, Tristan Naumann, Peter Szolovits, and John V. Guttag. Predicting Clinical Outcomes Across Changing Electronic Health Record Systems. In *KDD*, 2017.

Çaglar Gülçehre, Marcin Moczulski, Misha Denil, and Yoshua Bengio. Noisy Activation Functions. *CoRR*, abs/1603.00391, 2016.

Neil A Halpern and Stephen M Pastores. Critical Care Medicine Beds, Use, Occupancy, and Costs in the United States: A Methodological Review. *Critical care medicine*, 43 (11):2452–2459, 2015.

Hrayr Harutyunyan, Hrant Khachatrian, David C. Kale, Greg Ver Steeg, and Aram Galstyan. Multitask Learning and Benchmarking with Clinical Time Series Data. *Scientific Data*, 6(96), 2019.

Mahmud Hassan, Howard Tuckman, Robert Patrick, David Kountz, and Jennifer Kohn. Hospital Length of Stay and Probability of Acquiring Infection. *International Journal of Pharmaceutical and Healthcare Marketing*, 4:324–338, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CoRR*, abs/1512.03385, 2015.

Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural computation*, 9(8):1735–1780, 1997.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 448–456. JMLR, 2015.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aäron van den Oord, Alex Graves, and Koray Kavukcuoglu. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099, 2016.

Amit D. Kalra, Robert S Fisher, and Peter Axelrod. Decreased Length of Stay and Cumulative Hospitalized Days despite Increased Patient Admissions and Readmissions in an Area of Urban Poverty. *J Gen Intern Med.*, 25(9):920–935, 2010.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Kevin B. Laupland, Andrew W. Kirkpatrick, John B. Kortbeek, and Danny J. Zuege. Long-term Mortality Outcome Associated With Prolonged Admission to the ICU. *Chest*, 129(4):954 – 959, 2006.

Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network, 2013.

Zachary Chase Lipton, David C. Kale, Charles Elkan, and Randall C. Wetzel. Learning to Diagnose with LSTM Recurrent Neural Networks. *CoRR*, abs/1511.03677, 2015.

Gregory Mak, William D. Grant, James C McKenzie, and John B. Mccabe. Physicians' Ability to Predict Hospital Length of Stay for Patients Admitted to the Hospital from the Emergency Department. In *Emergency medicine international*, 2012.

Kusum S Mathews and Elisa F Long. A Conceptual Framework for Improving Critical Care Patient Flow and Bed Use. *Annals of the American Thoracic Society*, 12(6): 886–894, 2015.

Antonio Paulo Nassar and Pedro Caruso. ICU Physicians are Unable to Accurately Predict Length of Stay at Admission: A Prospective Study. *Journal of the International Society for Quality in Health Care*, 28 1:99–103, 2016.

Bret Nestor, Matthew B. A. McDermott, Geeticka Chauhan, Tristan Naumann, Michael C. Hughes, Anna Goldenberg, and Marzyeh Ghassemi. Rethinking Clinical Prediction: Why Machine Learning must Consider Year of Care and Feature Aggregation. *CoRR*, abs/1811.12583, 2018.

NHS Digital. DCB0084: OPCS-4.9 Requirements Specification, 2019.

Adam Paszke, Sam Gross, Francisco Massa, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

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

S. Purushotham, C. Meng, Z. Che, and Y. Liu. Benchmarking Deep Learning Models on Large Healthcare Datasets. *Journal of Biomedical Informatics*, 83:112–134, 2018.

Alvin Rajkomar, Eyal Oren, Kai Chen, et al. Scalable and Accurate Deep Learning with Electronic Health Records. In *npj Digital Medicine*, 2018.

John Rapoport, Daniel Teres, Yonggang Zhao, and Stanley Lemeshow. Length of Stay Data as a Guide to Hospital Economic Performance for ICU Patients. *Medical Care*, 41:386–397, 2003.

Seyedmostafa Sheikhalishahi, Vevake Balaraman, and Venet Osmani. Benchmarking Machine Learning Models on eICU Critical Care Dataset, 2019.

Benjamin Shickel, Tyler J. Loftus, Lasith Adhikari, Tezcan Ozrazgat-Baslanti, Azra Bihorac, and Parisa Rashidi. DeepSOFA: A Continuous Acuity Score for Critically Ill Patients using Clinically Interpretable Deep Learning. In *Scientific Reports*, 2019.

Huan Song, Deepta Rajan, Jayaraman J. Thiagarajan, and Andreas Spanias. Attend and Diagnose: Clinical Time Series Analysis using Attention Models. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 4091–4098. AAAI press, 2018.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15: 1929–1958, 2014.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020. URL https://distill.pub/2020/attribution-baselines/.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep net-

works. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 3319–3328. JMLR.org, 2017.

Harini Suresh, Nathan Hunt, Alistair E. W. Johnson, Leo Anthony Celi, Peter Szolovits, and Marzyeh Ghassemi. Clinical Intervention Prediction and Understanding with Deep Neural Networks. In *MLHC*, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *CoRR*, abs/1409.4842, 2014.

Nenad Tomašev, Xavier Glorot, Jack W Rae, et al. A Clinically Applicable Approach to Continuous Prediction of Future Acute Kidney Injury. *Nature*, 572(7767):116–119, 2019.

Sana Tonekaboni, Mjaye Mazwi, Peter Laussen, Danny Eytan, Robert Greer, Sebastian D. Goodfellow, Andrew Goodwin, Michael Brudno, and Anna Goldenberg. Prediction of Cardiac Arrest from Physiological Signals in the Pediatric ICU. In *MLHC*, 2018.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. *CoRR*, abs/1609.03499, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010. Curran Associates Inc., 2017.

David J. Wallace, Derek C. Angus, Christopher W. Seymour, Amber E. Barnato, and Jeremy M. Kahn. Critical Care Bed Growth in the United States. A Comparison of Regional and National Trends. *American Journal of Respiratory and Critical Care Medicine*, 191(4):410–416, 2015. PMID: 25522054.

Zhongyuan Wang, Peng Yi, Kui Jiang, Junjun Jiang, Zhen Han, Tao Lu, and Jiayi Ma. Multi-memory convolutional neural network for video super-resolution. *IEEE Transactions on Image Processing*, PP:1–1, 12 2018. doi: 10.1109/TIP.2018.2887017.

World Health Organisation. *ICD-10: International Statistical Classification of Diseases and Related Health Problems*, volume 10th Revision. World Health Organisation, 2011.

David Zimmerer, Jens Petersen, Gregor Köhler, Jakob Wasserthal, Tim Adler, Sebastian Wirkert, and Tobias Ross. trixi - Training and Retrospective Insight eXperiment Infrastructure. https://github.com/MIC-DKFZ/trixi, 2017.

Jack E Zimmerman, Andrew A Kramer, Douglas S McNair, and Fern M Malila. Acute Physiology and Chronic Health Evaluation (APACHE) IV: Hospital Mortality Assessment for Today's Critically Ill Patients. *Read Online: Critical Care Medicine | Society of Critical Care Medicine*, 34(5), 2006.

Appendix A. Model Architecture: Further Details

After N TPC layers have processed the time series, we apply two further pointwise convolutions to obtain the final predictions. Formally, these final steps (shown in Figure 6) can be written as

$$\underbrace{\hat{y}_t}_{(9)} = \text{HardTanh}\Big(\exp\Big(g'' * \underbrace{\sigma\Big(g' * \overbrace{\Big[\flat(\mathbf{h}_t^{(N)}) \parallel \mathbf{s} \parallel \mathbf{d}^*\Big]}^{\text{Final Combined In.(7)}}\Big)}_{\text{Penultimate Point. Out.(8)}}\Big)\Big) \tag{7}$$

where $B = R^{(N)} \times (Y+1) + S + D^*$ and the final pointwise filters are $g' : \{1, \ldots, B\} \rightarrow \mathbb{R}^{X \times 1}$ and $g'' : \{1, \ldots, X\} \rightarrow \mathbb{R}^{1 \times 1}$. Note that if a baseline model were to be used instead of TPC, the output dimensions would be H x 1 instead of B x 1, where H is the LSTM hidden size or d_{model} in the Transformer. We apply an exponential function to allow the upstream model to predict log(LoS) instead of LoS. We hypothesised that this could help to circumvent a common issue seen in previous models (e.g. Harutyunyan et al. (2019), as they struggle to produce predictions over the full dynamic range of length of stays). Finally, we apply a HardTanh function (Gülçehre et al., 2016) to clip any predictions that are smaller than 30 minutes or larger than 100 days, which protects against inflated MSLE loss values.

$$\text{HardTanh}(x) = \begin{cases} 100, & \text{if } x > 100, \\ \frac{1}{48}, & \text{if } x < \frac{1}{48}, \\ x, & \text{otherwise.} \end{cases} \tag{8}$$

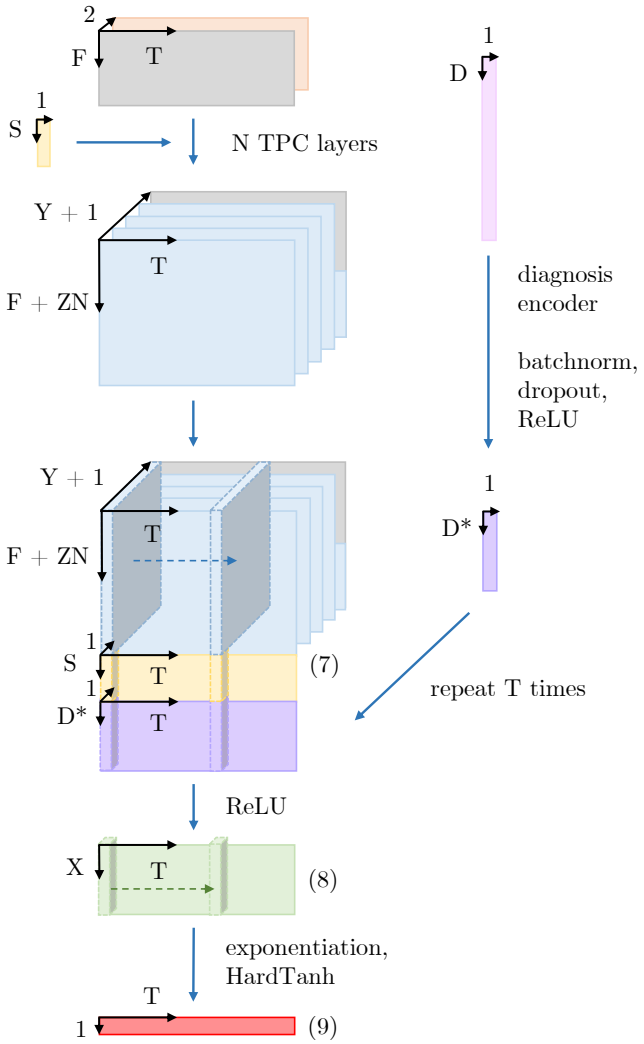

Figure 6: The original time series, $\mathbf{x}'$ (grey) and the decay indicators, $\mathbf{x}''$ (orange) are processed by N TPC layers before being combined with a diagnosis embedding $\mathbf{d}^*$ (purple) and static features $\mathbf{s}$ (yellow) along the feature axis. A two-layer pointwise convolution is applied to obtain the final predictions $\hat{\mathbf{y}}$ (red).

Appendix B. Feature Pre-processing

B.1. Static Features

The 17 static features are shown in Table 2. Discrete and continuous variables were scaled to the interval [-1, 1], using the 5th and 95th percentiles as the boundaries, and absolute cut offs were placed at [-4, 4]. This was to protect against large or erroneous inputs, while avoiding assumptions about the variable distributions. Binary variables were coded as 1 and 0. Categorical variables were converted to one-hot encodings.

Table 2: Static features. Age >89, Null Height and Null Weight were added as indicator variables to indicate when the age was more than 89 but has been capped, and when the height or weight were missing and have been imputed with the mean value.

Feature	**Type**	**Source Table**
Gender	Binary	*patient*
Age	Discrete	*patient*
Hour of Admission	Discrete	*patient*
Height	Continuous	*patient*
Weight	Continuous	*patient*
Ethnicity	Categorical	*patient*
Unit Type	Categorical	*patient*
Unit Admit Source	Categorical	*patient*
Unit Visit Number	Categorical	*patient*
Unit Stay Type	Categorical	*patient*
Num Beds Category	Categorical	*hospital*
Region	Categorical	*hospital*
Teaching Status	Binary	*hospital*
Physician Speciality	Categorical	*apachepatientresult*
Age >89	Binary	
Null Height	Binary	
Null Weight	Binary	

B.2. Diagnoses

Like many EHRs, diagnosis coding in eICU is hierarchical. At the lowest level they can be quite specific e.g. "neurologic | disorders of vasculature | stroke | hemorrhagic stroke | subarachnoid hemorrhage | with vasospasm". To maintain the hierarchical structure within a flat vector, we assigned separate features to each hierarchical level and use binary encoding. This produces a vector of size 4,436 with an average sparsity of 99.5% (only 0.5% of the data is positive). We apply a 1% prevalence cut-off on all these features to reduce the size of the vector to 293 and the average sparsity to 93.3%. If a disease does not make the cut-off for inclusion, it is still included via any parent classes that do make the cut-off (in the above example we record everything up to "subarachnoid hemorrhage"). We only included diagnoses that were recorded before the 5th hour in the ICU, to avoid leakage from the future.

Many diagnostic and interventional coding systems are hierarchical in nature: ICD-10 classification (World Health Organisation, 2011), Clinical Classifications Software (Elixhauser et al., 2015), SNOMED CT (De Silva et al., 2011) and OPCS Classification of Interventions and Procedures (NHS Digital, 2019), so this technique is generalisable to other coding systems present in EHRs.

B.3. Time Series

For each admission, 87 time-varying features (Table 3) were extracted from each hour of the ICU visit, and up to 24 hours before the ICU visit. The variables were processed in the same manner as the static features. In general, the sampling is very irregular, so the data was re-sampled according to one hour intervals and forward-filled. After forward-filling is complete, any data recorded before the ICU admission is removed. Decay indicators are added as described in Section 4.

Table 3: Time Series features. 'Time in the ICU' and 'Time of day' were not part of the tables in eICU but were added later as helpful indicators to the model.

Source Table			
lab			***respiratorycharting***
-basos	MPV	glucose	Exhaled MV
-eos	O2 Sat (%)	lactate	Exhaled TV (patient)
-lymphs	PT	magnesium	LPM O2
-monos	PT - INR	pH	Mean Airway Pressure
-polys	PTT	paCO2	Peak Insp. Pressure
ALT (SGPT)	RBC	paO2	PEEP
AST (SGOT)	RDW	phosphate	Plateau Pressure
BUN	WBC x 1000	platelets x 1000	Pressure Support
Base Excess	albumin	potassium	RR (patient)
FiO2	alkaline phos.	sodium	SaO2
HCO3	anion gap	total bilirubin	TV/kg IBW
Hct	bedside glucose	total protein	Tidal Volume (set)
Hgb	bicarbonate	troponin - I	Total RR
MCH	calcium	urinary specific gravity	Vent Rate
MCHC	chloride		
MCV	creatinine		
nursecharting	***vitalperiodic***	***vitalaperiodic***	**N/A**
Bedside Glucose	cvp	noninvasivediastolic	Time in the ICU
Delirium Scale/Score	heartrate	noninvasivemean	Time of day
Glasgow coma score	respiration	noninvasivesystolic	
Heart Rate	sao2		
Invasive BP	st1		
Non-Invasive BP	st2		
O2 Admin Device	st3		
O2 L/%	systemicdiastolic		
O2 Saturation	systemicmean		
Pain Score/Goal	systemicsystolic		
Respiratory Rate	temperature		
Sedation Score/Goal			
Temperature			

Appendix C. Hyperparameter Search Methodology and Implementation Details

Our model and its baselines have hyperparameters that can broadly be split into three categories: time series specific, non-time series specific and global parameters (shown in more detail in Tables 4, 5 and 6). The hyperparameter search ranges have been included in Table 7. First, we ran 25 randomly sampled hyperparameter trials on the TPC model to decide the non-time series specific parameters (diagnosis embedding size, final fully connected layer size, batch normalisation strategy and dropout rate) keeping all other parameters fixed. These parameters then remained fixed for all the models which share their non-time series specific architecture. We ran 50 hyperparameter trials to optimise the remaining parameters for the

TPC, standard LSTM, and Transformer models. To train the channel-wise LSTM and the temporal model with weight sharing, we ran a further 10 trials to re-optimise the hidden size (8 per feature) and number of temporal channels (32 channels shared across all features) respectively. For all other ablation studies and variations of each model, we kept the same hyperparameters where applicable (see Table 1 for a full list of all the models). The number of epochs was determined by selecting the best validation performance from a model trained over 50 epochs. This was different for each model: 8 for LSTM, 30 for CW LSTM, and 15 for the Transformer and TPC models. We noted that the best LSTM hyperparameters were similar to that found in Sheikhalishahi et al. (2019).

All deep learning methods were implemented in PyTorch (Paszke et al., 2019) and were optimised using Adam (Kingma and Ba, 2014). The data (including decay indicators) and the non-time series components of the models were the same as in TPC (Figure 6). We used trixi to structure our experiments and easily compare different hyperparameter choices (Zimmerer et al., 2017). The experiments were performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service (www.hpc.cam.ac.uk) funded by EPSRC Tier-2 capital grant EP/P020259/1.

Table 4: The TPC model has 11 hyperparameters (Main Dropout and Batch Normalisation have been repeated in the table because they apply to multiple parts of the model). We allowed the model to optimise a custom dropout rate for the temporal convolutions because they have fewer parameters and might need less regularisation than the rest of the model. The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

TPC Specific	
Temporal Specific	**Pointwise Specific**
Temp. Channels (12)	Point. Channels (13)
Temp. Dropout (0.05)	Main Dropout* (0.45)
Kernel Size (4)	
Batch Normalisation* (True)	
No. TPC Layers (9)	
Non-TPC Specific	**Global Parameters**
Diag. Embedding Size* (64)	Batch Size (32)
Main Dropout* (0.45)	Learning Rate (0.00226)
Final FC Layer Size* (17)	
Batch Normalisation* (True)	

Table 5: The LSTM model has 9 hyperparameters. We allowed the model to optimise a custom dropout rate for the LSTM layers. Note that batch normalisation is not applicable to the LSTM layers. The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

LSTM Specific	**Non-LSTM Specific**	**Global Parameters**
Hidden State (128)	Diag. Embedding Size* (64)	Batch Size (512)
LSTM Dropout (0.2)	Main Dropout* (0.45)	Learning Rate (0.00129)
No. LSTM Layers (2)	Final FC Layer Size* (17)	
	Batch Normalisation* (True)	

C.1. Transformer

Table 6: The Transformer model has 12 hyperparameters. We allowed the model to optimise a custom dropout rate for the Transformer layers. The positional encoding hyperparameter is binary; it determines whether or not we used the original positional encodings proposed by Vaswani et al. (2017). They were not found to be helpful (perhaps because we already have a feature to indicate the position in the time series (Section B.3)). Note that batch normalisation is not applicable to the Transformer layers (the default implementation uses layer normalisation). The best hyperparameter values are shown in brackets. Hyperparameters marked with * were fixed across all of the models.

Transformer Specific	**Non-Transformer Specific**	**Global Parameters**
No. Attention Heads (2)	Diag. Embedding Size* (64)	Batch Size (32)
Feedforward Size (256)	Main Dropout* (0.45)	Learning Rate (0.00017)
d_{model} (16)	Final FC Layer Size* (17)	
Positional Encoding (False)	Batch Normalisation* (True)	
Transformer Dropout (0)		
No. Transformer Layers (6)		

The Transformer is a multi-head self-attention model, originally designed for sequence-to-sequence tasks in natural language processing. It consists of both an encoder and decoder, however we only use the former because the LoS task is regression. Our implementation is the same as the original encoder in Vaswani et al. (2017), except that we add temporal masking to impose causality[9], and we omit the positional encodings because they were not helpful for the LoS task.

9. The processing of each timepoint can only depend on current or earlier positions in the sequence

Table 7: Hyperparameter Search Ranges. We took a random sample from each range and converted to an integer if necessary. For the kernel sizes (not shown in the table) the range was dependent on the number of TPC layers selected (because large kernel sizes combined with a large number of layers can have an inappropriately wide range as the dilation factor increases per layer). In general the range of kernel sizes was around 2-5 (but it could be up to 10 for small numbers of TPC Layers).

Hyperparameter	**Lower**	**Upper**	**Scale**
Batch Size	4	512	$\log_2$
Dropout Rate (all)	0	0.5	Linear
Learning Rate	0.0001	0.01	$\log_{10}$
Batch Normalisation	True	False	
Positional Encoding	True	False	
Diagnosis Embedding Size	16	64	$\log_2$
Final FC Layer Size	16	64	$\log_2$
Channel-Wise LSTM Hidden State Size	4	16	$\log_2$
Point. Channels	4	16	$\log_2$
Temp. Channels	4	16	$\log_2$
Temp. Channels (weight sharing)	16	64	$\log_2$
LSTM Hidden State Size	16	256	$\log_2$
d_{model}	16	256	$\log_2$
Feedforward Size	16	256	$\log_2$
No. Attention Heads	2	16	$\log_2$
No. TPC Layers	1	12	Linear
No. LSTM Layers	1	4	Linear
No. Transformer Layers	1	10	Linear

Appendix D. Evaluation Metrics

The metrics we use are: mean absolute deviation (MAD), mean absolute percentage error (MAPE), mean squared error (MSE), mean squared loss error (MSLE), coefficient of determination (R^2) and Cohen Kappa Score. We modify the MAPE metric slightly so that very small true LoS values do not produce unbounded MAPE values. We place a 4 hour lower bound on the divisor i.e.

$$\text{Absolute Percentage Error} = \left| \frac{y_{true} - y_{pred}}{\max\left(y_{true}, \frac{4}{24}\right)} \right| * 100$$

MAD and MAPE are improved by centering predictions on the median. Likewise, MSE and R^2 are bettered by centering predictions around the mean. They are more affected by the skew. MSLE is a good metric for this task, indeed, it is the loss function in most experiments, but is less readily-interpretable than some of the other measures. Cohen's linear weighted Kappa Score (Cohen, 1960) is intended for ordered classification tasks rather than regression, but it can effectively mitigate for skew if the bins are chosen well. It has previously provided useful insights in Harutyunyan et al. (2019), so we use the same LoS bins: 0-1, 1-2, 2-3, 3-4, 4-5, 5-6, 6-7, 7-8, 8-14, and 14+ days. As a classification measure, it will treat everything

falling within the same classification bin as equal, so it is fundamentally a coarser measure than the other metrics.

To illustrate the importance of using multiple metrics, consider that the mean and median models are in some sense equally poor (neither has learned anything meaningful for our purposes). Nevertheless, the median model is able to better exploit the MAD, MAPE and MSLE metrics, and the mean model fares better with MSE, but the Kappa score betrays them both. A good model will perform well across all of the metrics.

Appendix E. Additional Investigations

E.1. Feature Importance

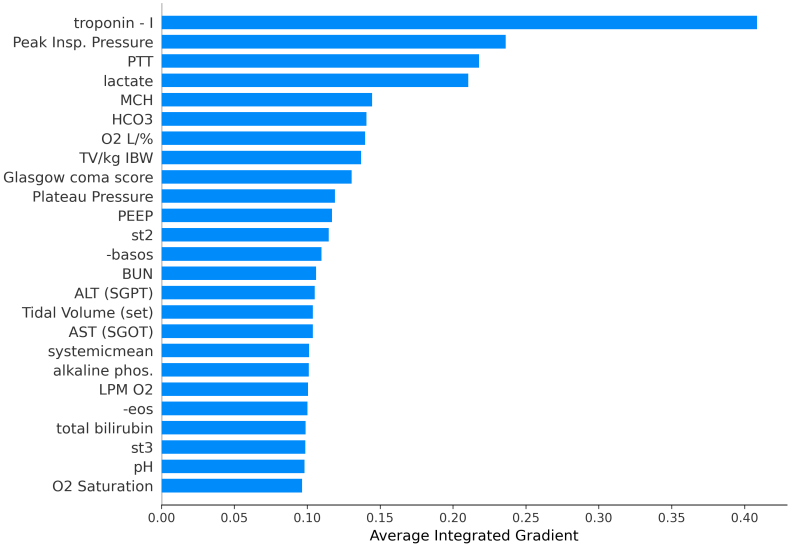

Figure 7: Top 25 most important features according to the mean absolute integrated gradient feature attribution values when a single LoS prediction is made at 24 hours per patient.

We used the integrated gradients method (Sundararajan et al., 2017) to calculate feature attributions for the LoS estimates. This method computes the importance scores ϕ_i^{IG} by accumulating gradients interpolated between a baseline $\mathbf{b}$ input (intended to represent the absence of data) and the current input $\mathbf{x}$.

$$\phi_i^{IG}(\psi, \mathbf{x}, \mathbf{b}) = \overbrace{(\mathbf{x}_i - \mathbf{b}_i)}^{\text{diff. from baseline}} \times \int_{\alpha=0}^{1} \overbrace{\frac{\delta\psi(\mathbf{b} + \alpha(\mathbf{x} - \mathbf{b}))}{\delta \mathbf{x}_i}}^{\text{acc. local grad.}} d\alpha \tag{9}$$

where the TPC model is represented as ψ. The background and intuition behind the method is explained clearly by Sturmfels et al. (2020). We use the mean feature values as our baseline input vector.

Analysing Figure 7, we note that the top features are all strong indicators of organ failure. Troponin - I is a specific biomarker of myocardial infarction. Peak insp. pressure, O2 L/%, TV/kg IBW, Plateau Pressure, PEEP and Tidal Volume (set) would all indicate that the

patient is on a respiratory ventilator (on account of respiratory failure). PTT, ALT (SGPT), AST (SGOT) and alkaline phosphatase all suggest liver disease, and high BUN or bilirubin levels point toward kidney failure. Finally we see infection markers: lactate, -basos and -eos which could indicate sepsis. Both multi-organ failure and sepsis are known causes of extended LoS in the ICU (Böhmer et al., 2014).

E.2. Time Series Ablation

Table 8: Performance of the TPC model and its baselines when only some of the time series are included. The indicator '(labs)' means that the model has been trained exclusively on laboratory tests, '(other)' refers to everything except labs: vital signs, nurse observations and machine logged variables. The metric acronyms, colour scheme and confidence interval calculation is described in the legend to Table 1. The percentage impairment when compared to the complete dataset is shown in grey underneath the absolute values. They are calculated with respect to the best value for the metric: 0 for MAD, MAPE, MSE and MSLE, and 1 for R^2 and Kappa. A large percentage impairment means that the model does much better with complete data i.e. it has a high 'percentage gain' from the combination of both data types compared to the ablation case.

Model	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM	**2.39±0.00**	**118.2±1.1**	**26.9±0.1**	**1.47±0.01**	**0.09±0.00**	**0.28±0.00**
LSTM (labs)	2.43±0.00	123.8±1.2	27.3±0.1	1.57±0.00	0.08±0.00	0.27±0.00
	(-1.7%)	(-4.7%)	(-1.5%)	(-6.8%)	(-1.1%)	(-1.4%)
LSTM (other)	2.41±0.00	120.2±0.7	27.3±0.1	1.49±0.00	0.07±0.00	0.27±0.00
	(-0.8%)	(-1.7%)	(-1.5%)	(-1.4%)	(-2.2%)	(-1.4%)
CW LSTM	**2.37±0.00**	**114.5±0.4**	**26.6±0.1**	**1.43±0.00**	**0.10±0.00**	**0.30±0.00**
CW LSTM (labs)	2.42±0.00	124.4±0.7	27.0±0.1	1.57±0.00	0.08±0.00	0.28±0.00
	(-2.1%)	(-8.6%)	(-1.5%)	(-9.8%)	(-2.2%)	(-2.9%)
CW LSTM (other)	2.41±0.00	120.6±0.8	27.1±0.1	1.51±0.00	0.08±0.00	0.29±0.00
	(-1.7%)	(-5.3%)	(-1.9%)	(-5.6%)	(-2.2%)	(-1.4%)
Transformer	**2.36±0.00**	**114.1±0.6**	**26.7±0.1**	**1.43±0.00**	**0.09±0.00**	**0.30±0.00**
Transformer (labs)	2.42±0.00	121.0±0.7	27.3±0.1	1.56±0.00	0.07±0.00	0.27±0.00
	(-2.5%)	(-6.0%)	(-2.2%)	(-9.1%)	(-2.2%)	(-4.3%)
Transformer (other)	2.40±0.00	118.3±0.6	27.3±0.1	1.50±0.00	0.07±0.00	0.27±0.00
	(-1.7%)	(-3.7%)	(-2.2%)	(-4.9%)	(-2.2%)	(-4.3%)
TPC	**1.78±0.02**	**63.5±4.3**	**21.7±0.5**	**0.70±0.03**	**0.27±0.02**	**0.58±0.01**
TPC (labs)	1.85±0.01	72.0±2.2	22.5±0.2	0.81±0.01	0.24±0.01	0.55±0.00
	(-3.9%)	(-13.4%)	(-3.7%)	(-15.7%)	(-4.1%)	(-7.1%)
TPC (other)	**1.81±0.02**	**68.5±4.7**	**21.8±0.3**	0.77±0.03	**0.26±0.01**	**0.57±0.01**
	(-1.7%)	(-7.9%)	(-0.5%)	(-10.0%)	(-1.4%)	(-2.4%)

We performed ablations on the type of time series variable that we include: laboratory tests only (labs), which are infrequently sampled, and all other variables (other) which include vital signs, nurse observations, and automatically recorded variables (e.g. from ventilator machines). This shows how well each model can cope with time series exhibiting different periodicity and sampling frequencies. The results are shown in Table 8.

The TPC model has the largest percentage gain when the labs and other variables are combined (this is synonymous with the greatest percentage impairment in the ablations). Next are the CW LSTM and Transformer, followed by the LSTM. This suggests that the TPC model is best able to exploit EHR time series with different temporal properties.

When examining the results for LSTM and CW LSTM in more detail, we can see that the CW LSTM only has an advantage when the model has to combine the data types. This supports the hypothesis that the CW LSTM is better able to cope when there are varying frequencies in the data, as it can tailor the processing to each. When the inter-feature variability is small (the same type of time series) they perform similarly.

It is unsurprising that the Transformer does better than the LSTM when combining data types, as it can directly skip over large gaps in time to extract a trend in lab values, while simultaneously attending to recent timepoints for the processing of other variables.

The TPC is the most successful model; its inherent periodic structure helps it to extract useful information from all of the variables. The CW LSTM and Transformer do not have this in their architectures, making the derivation more obscure. The importance of periodicity is discussed in more detail in Section 6.

E.3. Training Data Size

Researchers may wish to know how large their training data should be to benefit from using the TPC model. We tested the TPC, LSTM, CW LSTM, and Transformer models with 6.25%, 12.5%, 25%, 50%, and 100% of the training data. TPC maintains the best test performance on all data sizes, with an increasing benefit for larger data. Figure 8 shows the effect on MSLE. Table 9 shows the full results for all metrics.

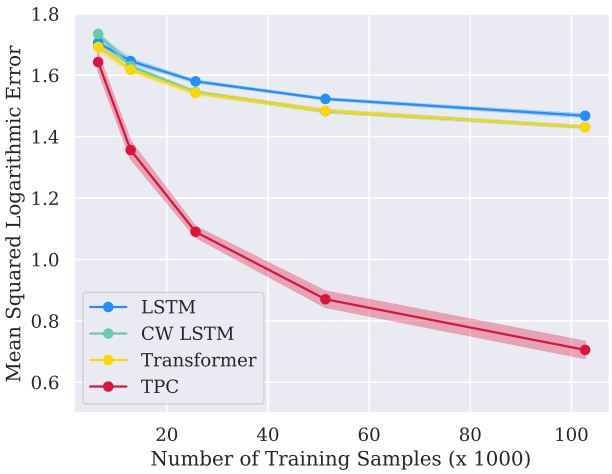

Figure 8: The effect of changing the size of the training data on the LSTM, CW LSTM, Transformer, and TPC model performance. Only the mean squared logarithmic error (MSLE) is shown for clarity, however the other metrics are shown in Table 9. Note that the performance of the CW LSTM and Transformer models are so similar that the curves are superimposed.

Table 9: The effect of changing the size of the training data on the LSTM, CW LSTM, Transformer, and TPC model performance. A hundred percent of the training set represents 102,712 ICU stays, 50% is 51,356, 25% is 25,678, 12.5% is 12,839, and 6.25% is 6,420 stays.

Model (% train data)	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM (100)	2.39±0.00	118.2±1.1	26.9±0.1	1.47±0.01	0.09±0.00	0.28±0.00
LSTM (50)	2.41±0.01	129.9±1.9	26.2±0.2	1.52±0.00	0.11±0.01	0.31±0.01
LSTM (25)	2.44±0.01	126.8±2.5	27.2±0.3	1.58±0.00	0.08±0.01	0.27±0.01
LSTM (12.5)	2.48±0.01	137.4±3.4	27.4±0.2	1.65±0.01	0.07±0.01	0.27±0.01
LSTM (6.25)	2.52±0.02	135.9±3.3	28.0±0.8	1.71±0.02	0.05±0.03	0.26±0.03
CW LSTM (100)	2.37±0.00	114.5±0.4	26.6±0.1	1.43±0.00	0.10±0.00	0.30±0.00
CW LSTM (50)	2.40±0.01	123.4±0.7	26.5±0.1	1.48±0.01	0.10±0.00	0.31±0.00
CW LSTM (25)	2.44±0.00	119.8±1.3	27.2±0.1	1.54±0.00	0.08±0.00	0.29±0.00
CW LSTM (12.5)	2.50±0.01	134.7±1.5	27.7±0.1	1.63±0.01	0.06±0.00	0.28±0.00
CW LSTM (6.25)	2.58±0.01	129.8±3.5	29.0±0.2	1.73±0.01	0.02±0.01	0.25±0.01
Transformer (100)	2.36±0.00	114.1±0.6	26.7±0.1	1.43±0.00	0.09±0.00	0.30±0.00
Transformer (50)	2.39±0.00	120.1±0.6	26.5±0.1	1.48±0.00	0.10±0.00	0.31±0.00
Transformer (25)	2.43±0.01	117.9±1.8	27.2±0.2	1.54±0.01	0.08±0.01	0.28±0.01
Transformer (12.5)	2.48±0.01	128.1±2.3	27.9±0.1	1.62±0.01	0.06±0.00	0.26±0.01
Transformer (6.25)	2.52±0.01	139.7±2.4	27.8±0.1	1.69±0.02	0.06±0.00	0.26±0.00
TPC (100)	1.78±0.02	63.5±4.3	21.7±0.5	0.70±0.03	0.27±0.02	0.58±0.01
TPC (50)	1.95±0.02	72.0±3.1	23.8±0.4	0.87±0.03	0.19±0.01	0.51±0.01
TPC (25)	2.09±0.01	89.0±3.8	24.8±0.3	1.09±0.02	0.16±0.01	0.45±0.01
TPC (12.5)	2.28±0.01	101.4±4.8	27.0±0.4	1.36±0.03	0.08±0.01	0.35±0.02
TPC (6.25)	2.49±0.02	139.9±5.5	28.0±0.3	1.64±0.03	0.05±0.01	0.28±0.01

E.4. Loss Function

Table 10: The effect of training with the mean squared logarithmic error (MSLE) loss function when compared to mean squared error (MSE). This is an extension to Table 1 (refer to its legend for definitions of the metric acronyms, detailed of CI calculations and meaning of the colour scheme).

Model	MAD	MAPE	MSE	MSLE	R^2	Kappa
LSTM (MSLE)	**2.39±0.00**	**118.2±1.1**	26.9±0.1	**1.47±0.01**	0.09±0.00	**0.28±0.00**
LSTM (MSE)	2.57±0.03	235.2±6.2	**24.5±0.2**	1.97±0.02	**0.17±0.01**	**0.28±0.01**
CW LSTM (MSLE)	**2.37±0.00**	**114.5±0.4**	26.6±0.1	**1.43±0.00**	0.10±0.00	0.30±0.00
CW LSTM (MSE)	2.56±0.01	218.5±4.0	**24.2±0.1**	1.84±0.02	**0.18±0.00**	**0.34±0.01**
Transformer (MSLE)	**2.36±0.00**	**114.1±0.6**	26.7±0.1	**1.43±0.00**	0.09±0.00	**0.30±0.00**
Transformer (MSE)	2.51±0.01	212.7±5.2	**24.7±0.2**	1.87±0.03	**0.16±0.01**	0.28±0.01
TPC (MSLE)	**1.78±0.02**	**63.5±4.3**	**21.7±0.5**	**0.70±0.03**	**0.27±0.02**	**0.58±0.01**
TPC (MSE)	2.21±0.02	154.3±10.1	**21.6±0.2**	1.80±0.10	**0.27±0.01**	0.47±0.01

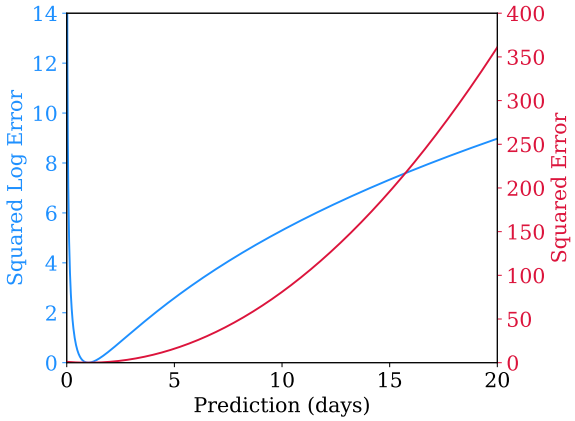

Figure 9: The behaviour of squared logarithmic error (blue) and squared error (red) functions when the true LoS is 1 day.

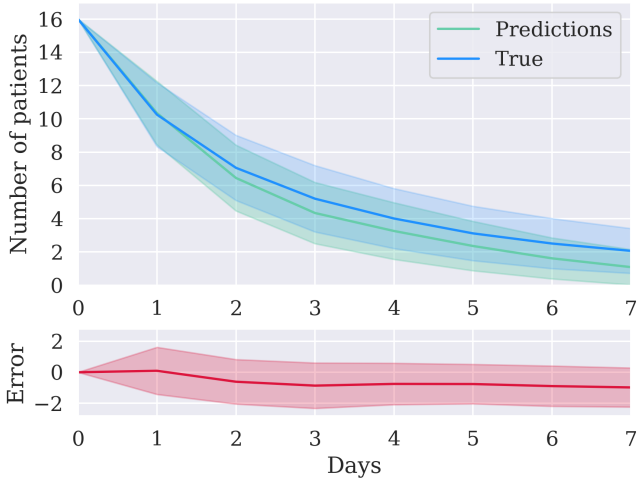

Figure 10: ICU simulation. We show the number of patients remaining in the ICU over time from an initial cohort of 16 patients from 500 random simulations. The shaded regions show the standard deviation across the runs. Error is calculated from True minus Predictions.

Appendix F. ICU Simulation

From the perspective of a bed manager, *aggregate* performance of the model is important. To investigate this, we performed a simulation study. We ran 500 ICU simulations by randomly selecting 16 examples from the test set to form a 'virtual cohort'. The number 16 was chosen because US hospitals have, on average, 24 ICU beds (Wallace et al., 2015) with an occupancy rate of 68% (Halpern and Pastores, 2015). Figure 10 shows the number of patients remaining in the ICU (of the selected cohort; we do not visualise incoming ICU admissions) using their true remaining LoS (blue). We compute the error (red) between the predictions (green) and true values. The model is well calibrated when predicting patients who are going to stay for at least 1 day. After this, the model tends to under-predict the occupancy by approximately 0.8 patients, corresponding to a small bias towards under-estimating the remaining LoS.