Dynamic outcomes-based clustering of disease progression in mechanically ventilated patients

Anonymous Author(s)
Affiliation
Address
email

Abstract

The advancement of Electronic Health Records (EHRs) and machine learning have enabled a data-driven and personalised approach to healthcare. One step in this direction is to uncover patient sub-types with similar disease trajectories in a heterogeneous population. This is especially important in the context of mechanical ventilation in intensive care, where mortality is high and there is no consensus on treatment. In this work, we present a new approach to clustering mechanical ventilation episodes, using a multi-task combination of supervised, self-supervised and unsupervised learning techniques. Our dynamic clustering assignment is explicitly guided to reflect the *phenotype*, *trajectory* and *outcomes* of the patient. Experimentation on a real-world dataset is encouraging, and we hope that we could someday translate this into actionable insights in guiding future clinical research.

1 Introduction

Patients on mechanical ventilation are a highly heterogeneous group, with widely differing outcomes. Some have relatively healthy lungs e.g. if they are recovering from surgery on another organ; whereas others have varying degrees of pulmonary failure. Pulmonary failure can be acute e.g. Acute Respiratory Distress Syndrome (ARDS) and deteriorate rapidly, or chronic, typically evolving slowly. Unfortunately, patients on ventilators have high mortality [16, 19] and there is no established consensus on optimal treatment strategies from randomised controlled trials [1]. Therefore, *there is significant potential benefit of a data-driven strategy to guide future clinical studies*.

To this end, we have developed a dynamic clustering approach for mechanically ventilated patients in the Intensive Care Unit (ICU). We use a mixture of supervised, self-supervised and unsupervised techniques to reveal key subgroups in the Amsterdam UMC critical care dataset. Our clusters are designed to share similarities in *phenotype*, *trajectory* and *outcomes*. The assignment is dynamic, so we generate a cluster for each hour that a patient remains in the ICU. This means that if an event happens which alters the predicted trajectory and outcomes, we will see a shift in the cluster assignment. This is interesting, not only because it can reveal which events are associated with these shifts, but it may allow us to gain insight into what might have happened if the ventilation strategy had been different. Our hope is that it improve our understanding of lung disease, bring us close to an individualised approach to mechanical ventilation.

2 Background

Temporal clustering is the unsupervised process of partitioning time series into homogeneous groups. When applied to clinical data, it can be challenging because the time series are not the same length, and the data contains significant noise (both in the underlying physiology and in the recording

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

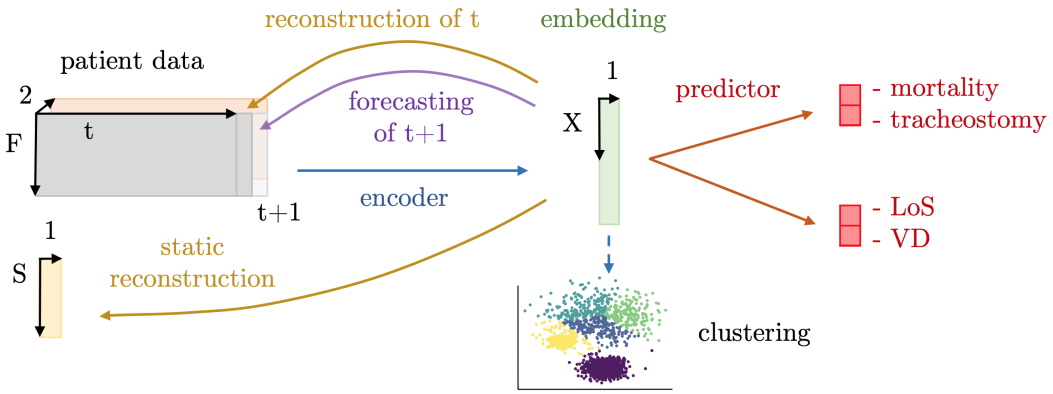

Figure 1: Overview of our model. Only one timestep, t, is shown for simplicity. F is the number of time series features, T is the time series length and S is the number of static variables. At timestep t, the static variables (yellow) and preceeding time series variables (grey) and their corresponding decay indicator variables (orange, explained under 'Time Series' in A.5) are given to the encoder, which produces an embedding (green) for timestep t. This embedding is then given to the decoder networks (yellow), forecasting network (purple) and the predictor network to obtain the four patient outcomes (red), outlined in Section 3. After training is complete, the test embeddings are used for clustering.

process). Temporal clustering approaches been done before e.g. in Parkinsons [31], diabetes [22] and cystic fibrosis [13] however data-driven clustering has not been comprehensively undertaken for mechanical ventilation.

3 Methods

Broadly, our strategy is to train a temporal encoder to embed the patient data at every timestep (this is analogous to returning all the hidden states for an LSTM model). We use a mixture of supervised, unsupervised and self-supervised learning to do this. Once training is complete, we then use an unsupervised method to cluster the embeddings, so that we get a cluster for every timestep in the patient's ventilation episode. Our code can be found at `LINK_TO_BE_RELEASED`.

The data consists of both timeseries and static features (see Section 4). Our supervised tasks include two binary tasks: hospital mortality and the risk of receiving a tracheostomy[1], and two duration tasks: the remaining length of stay (LoS) from timestep t, and their remaining ventilation duration (VD). This ensures that the patient *outcomes* are stored within the embedding. In addition, we train a decoder to reconstruct timestep t and the static data. This is an unsupervised approach which encourages the embedding to retain the patient *phenotype*. Finally, we predict timestep $t + 1$, a self-supervised approach designed to embed the patient *trajectory*. See Figure 1 for a schematic.

Encoder models In recent years, LSTMs have been by far the most popular model for predicting clinical outcomes and have achieved state-of-the-art results [7, 23, 20]. They have also been applied to other patient prediction tasks e.g. forecasting diagnoses and medications [4, 15], and mortality prediction [3, 7, 24]. More recently, the Transformer model [30] has marginally outperformed the LSTM when predicting LoS [25]. However most recently, the Temporal Pointwise Convolution (TPC) model [21] has outperformed both the LSTM and Transformer models on mortality and LoS. The TPC works by embedding clinical trajectories by convolving across time and across features. See Section A.1 for more detail.

Clustering We use a kmeans clustering strategy to cluster the learned embeddings (more detail in Section A.7). We determined that euclidean distance is a more sensible choice than cosine similarity for pairwise comparisons because intuitively, it is not just the fact that the patient is deteriorating in a certain direction that should matter, but also the magnitude of that deterioration. As detailed in Section A.1, we applied batch normalisation [10] to the embeddings, to ensure that the embedding distribution remained within a reasonable range.

[1] A tracheostomy is a procedure designed for long term mechanical ventilation of a patient.

Table 1: Performance of the various encoders on the prediction tasks averaged over 5 independent training runs. The error margins are 95% confidence intervals. For mortality and tracheostomy we report the area under the receiver operating characteristic curve (AUROC) and the area under the precision recall curve (AUPRC). For LoS and VD we report the mean absolute deviation (MAD) (the primary metric used in Harutyunyan et al. [7]) and mean squared log error (MSLE) (the recommended metric in Rocheteau et al. [21]). (a) is the full multi-task setting, (b) is prediction-task only and (c) is a variational alternative (to the full task setting). The best results are highlighted in blue. If the result is statistically significant on a t-test then it is indicated with daggers († = p < 0.05, ‡ = p < 0.001).

		In-Hospital Mortality		**Tracheostomy**		**Length of Stay**		**Ventilation Duration**	
	Model	**AUROC**	**AUPRC**	**AUROC**	**AUPRC**	**MAD**	**MSLE**	**MAD**	**MSLE**
(a)	TPC	**0.828±0.004**‡	**0.643±0.005**‡	**0.798±0.003**‡	**0.477±0.006**‡	**7.20±0.05**‡	**0.366±0.008**‡	**3.40±0.05**‡	**0.224±0.010**‡
	Transformer	0.705±0.016	0.438±0.007	0.724±0.011	0.432±0.008	8.27±0.11	0.496±0.006	3.60±0.04	0.246±0.008
	LSTM	0.813±0.003	0.596±0.005	0.751±0.004	0.402±0.005	9.27±0.03	0.681±0.002	5.60±0.02	0.674±0.005
(b)	TPC	**0.829±0.003**‡	**0.645±0.005**‡	**0.796±0.003**‡	0.475±0.006	**7.47±0.03**‡	**0.375±0.008**‡	**3.21±0.03**‡	**0.207±0.009**‡
	Transformer	0.694±0.014	0.437±0.010	0.736±0.011	0.395±0.009	8.01±0.08	0.477±0.006	3.71±0.04	0.247±0.008
	LSTM	0.820±0.004	0.603±0.006	0.774±0.005	**0.474±0.005**	9.12±0.03	0.676±0.002	5.61±0.02	0.697±0.006
(c)	TPC	**0.815±0.003**‡	**0.620±0.005**‡	**0.771±0.004**‡	**0.427±0.004**‡	**8.78±0.02**‡	**0.486±0.008**‡	**4.06±0.03**‡	**0.287±0.005**‡
	Transformer	0.635±0.016	0.334±0.007	0.699±0.011	0.345±0.008	9.46±0.11	0.636±0.006	4.76±0.04	0.390±0.008
	LSTM	0.782±0.004	0.526±0.006	0.735±0.005	0.407±0.005	9.96±0.03	0.800±0.002	5.94±0.02	0.785±0.006

4 Data

We used the Amsterdam UMC database version 1.0.2, which contains 23,106 ICU admissions from 20,109 patients admitted between 2003 and 2016. We selected all the mechanical ventilation episodes with a minimum duration of 4 hours, capping the maximum duration after 21 days to reduce computational costs. This corresponded to 14,836 episodes which occurred during 13,502 ICU admissions from a cohort of 12,597 unique patients. We selected 31 time series features from the *numericitems* table. To help the model cope with this missing data, we forward-filled the data in the gaps (this is the more realistic than interpolation as the clinician would only have access to the most recent value). We then added 'decay indicators' to specify where the data is stale. The decay was calculated as 0.8^j, where j is the time since the last recording, very similar to that in Rocheteau et al. [21]. We also extracted 14 static features from the *admissions* table (see Tables 2 and 3, and Appendix A.5 for the full list of features and further details). The data were split such that 70%, 15% and 15% were used for training, validation and testing respectively. This was split by *patient*, not ventilation episode, to avoid data leakage from the train set.

5 Results

Task Performance Looking at Table 1, we have established that the TPC model performs significantly better than the LSTM and Transformer in almost all settings on the Amsterdam data, which is in line with previous findings in MIMIC-IV and eICU. Interestingly, the Transformer appears to perform poorly on the binary tasks but better on the duration tasks. Another curiosity is that the LSTM performs the best on the self-supervised and unsupervised auxillary tasks. These results are briefly discussed in Section B.1.

Predictor Only Task Variant We performed an ablation study to investigate whether the inclusion of the unsupervised and self-supervised tasks (Table 1a) would hurt the performance on the prediction tasks compared to the predictor only scenario (Table 1b). In the full multi-task setting, the model now has competing learning objectives. The results are indeed mostly better in the predictor only scenario, however, the TPC was able to maintain very similar performance in the full multi-task setting.

Variational Embedding Spaces We experimented with making the embeddings "variational", by representing the means and standard deviations for the embedding coordinates and sampling. The

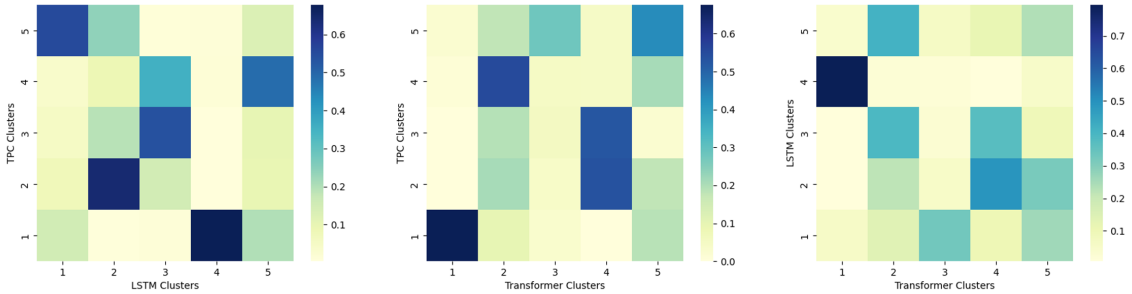

Figure 2: A comparison of the cluster assignments produced by different encoders. We can see that for the most part there is a distinct mapping between clusters i.e. cluster 1 in the TPC corresponds almost perfectly to cluster 4 in the LSTM.

idea was that by forcing the embedding space to be smoother, we might improve the quality of the clustering as the distances between patients in the embedding space become more reliable. However, this was found to universally hurt performance (Table 1c).

Cluster Analysis Figure 2 shows a map of the timepoints in the test set, comparing which cluster they were assigned in different models. It is encouraging that there is a strong cohesion between the different encoders, meaning that the models are picking out genuine consistent patterns in the data. Further cluster analysis including outcome statistics for each cluster can be found in Section B.2. Taken together these early results suggest the clusters are well correlated with outcomes and are clinically meaningful.

6 Limitations and Future Work

We are actively working on several strategies to improve the clusters that we have presented here.

Hierarchical Clustering It is evident from the results that certain clusters are more related than others. A tree based hierarchy of clusters seems more natural than a flat structure. It will be interesting to visualise the shifts in cluster assignment within a tree rather than as discrete jumps. We are currently working on a modification of an approach being used for genetics data [18, 2, 5], and are hoping to apply it to the ICU.

Contrastive Learning We are investigating the use of contrastive learning to regularise the embedding space (more detail in Section B.3). Currently, there is no explicit loss to enforce similarity in the embedding space to be linearly related to differences in the raw data. Despite this, we have empirically found the clusters presented in this paper to be plausible clinically. This is likely to be because our predictor and decoder networks are very simple, meaning that the embedding space does not have the freedom to model similar patient trajectories (which should be in the same cluster) in disparate parts of the embedding space. Nevertheless, contrastive learning could provide further regularisation to reduce the chance of any unexpected behaviour.

Cluster Reliability We would like to demonstrate that the approach is generalisable. We have already shown that the results with different encoders is comparable, but repeating the work on another dataset, such as MIMIC-IV or eICU would strengthen this assessment. We would also like to do further analyses on the clusters to generate hypotheses for future clinical studies.

While we acknowledge these limitations, we have shown that there could be merit in this approach from the initial results presented in this paper. Firstly, we have repeated the previously known result that the TPC model outperforms LSTM and Transformer but in a new dataset. We have combined supervised, self-supervised and unsupervised learning to improve the robustness of the embeddings. Most importantly, we have shown that three different encoders were able to produce clinically meaningful clusters from their embeddings alone, and were consistent in the features that they isolated. We look forward to improving the approach further in the coming weeks.

References

[1] Thomas Bein et al. The standard of care of patients with ARDS: ventilatory settings and rescue therapies for refractory hypoxemia. *Intensive Care Medicine*, 42(5):699–711, 2016.

[2] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *CoRR*, abs/2010.00402, 2020. URL `https://arxiv.org/abs/2010.00402`.

[3] 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.

[4] 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.

[5] Gabriele Corso, Rex Ying, Michal Pándy, Petar Veličković, Jure Leskovec, and Pietro Liò. Neural distance embeddings for biological sequences, 2021. URL `https://arxiv.org/abs/2109.09740`.

[6] WA Falcon. Pytorch lightning. *GitHub*, 2019. URL `https://github.com/PyTorchLightning/pytorch-lightning`.

[7] 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.

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

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

[10] 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.

[11] 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.

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

[13] Changhee Lee and Mihaela van der Schaar. Temporal phenotyping using deep predictive clustering of disease progression, 2020. URL `https://arxiv.org/abs/2006.08600`.

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

[15] 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.

[16] Jan Máca et al. Past and present ards mortality rates: A systematic review. *Respiratory Care*, 62(1):113–122, 2017.

[17] 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.

[18] Aman Patel, Daniel Mas Montserrat, Carlos Bustamante, and Alexander Ioannidis. Hyperbolic geometry-based deep learning methods to produce population trees from genotype data. *bioRxiv*, 2022. doi: 10.1101/2022.03.28.484797. URL `https://www.biorxiv.org/content/early/2022/03/29/2022.03.28.484797`.

[19] J. Poole et al. Individual patient data analysis of tidal volumes used in three large randomized control trials involving patients with acute respiratory distress syndrome. *BJA: British Journal of Anaesthesia*, 118(4):570–575, 2017.

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

[21] Emma Rocheteau, Pietro Liò, and Stephanie Hyland. Temporal pointwise convolutional networks for length of stay prediction in the intensive care unit. In *Proceedings of the Conference on Health, Inference, and Learning*, CHIL '21, page 58–68, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383592. doi: 10.1145/3450439.3451860. URL `https://doi.org/10.1145/3450439.3451860`.

[22] Alexander Rusanov, Patric V. Prado, and Chunhua Weng. Unsupervised time-series clustering over lab data for automatic identification of uncontrolled diabetes. In *2016 IEEE International Conference on Healthcare Informatics (ICHI)*, pages 72–80, 2016. doi: 10.1109/ICHI.2016.14.

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

[24] 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.

[25] 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.

[26] 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.

[27] 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.

[28] Patrick Thoral, Ronald Driessen, and Tariq Dam. Amsterdamumcdb. *GitHub*, 2020. URL `https://github.com/AmsterdamUMC/AmsterdamUMCdb`.

[29] 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.

[30] 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.

[31] Xi Zhang, Jingyuan Chou, Jian Liang, Cao Xiao, Yize Zhao, Harini Sarva, Claire Henchcliffe, and Fei Wang. Data-Driven Subtyping of Parkinson's Disease Using Longitudinal Clinical Records: A Cohort Study. *Scientific Reports*, 9(1):797, 2019. ISSN 2045-2322. doi: 10.1038/s41598-018-37545-z. URL `https://doi.org/10.1038/s41598-018-37545-z`.

A Implementation Details

A.1 TPC Model Architecture

We use a Temporal Pointwise Convolution (TPC) [21] network as one of our encoders. This is a model that takes advantage of both temporal convolution (to analyse trends) and pointwise convolution (to look for any important variable interactions). It is inspired by the way that clinicians would approach an assessment of a patient e.g. they might check how the respiratory rate is changing over time, and they may also look at combination features e.g. the PaO_2/FiO_2 ratio. The components of the network are briefly explained below.

Temporal Convolution Temporal Convolution Networks (TCNs) [29, 11] are models that convolve over the time dimension. The TPC model uses stacked TCNs to extract *temporal trends* in the data. Unlike most implementations, it *does not share weights across features* i.e. weight sharing is only across time. This is because the features in a typical EHR differ sufficiently in their temporal characteristics and warrant specialised processing. In TCNs, the receptive field sizes[2] are highly adaptable. They can be increased by using greater dilation, larger kernel sizes or by stacking more layers. By contrast, recurrent neural networks such as LSTM can only process one time step at a time, and Transformers have a weaker sense of temporal structure e.g. periodicity, which is central to understanding time series in the EHR.

Pointwise Convolution Pointwise (or 1x1) convolution [14] is typically used to reduce the dimensions in an input [27]. However in the TPC model it is used to compute *interaction features* from the existing feature set at each timepoint.

Skip Connections Skip connections [8] allow each layer to see the original data and the pointwise outputs from previous layers. This helps the network to cope with infrequently sampled data.

Temporal Pointwise Convolution The full model combines temporal and pointwise convolution in parallel. Figure 3 shows just one layer, however our implementation has 6 layers stacked sequentially (Table 4). With each successive TPC layer, the temporal dilation is increased by 1.

After processing the time series, the output is combined with static (non time-varying) features (Figure 4 shows the complete architecture). To obtain the embedding, the combined features are passed through a small two-layer pointwise convolution. The predictor networks are then only one layer, with a sigmoid activation for the binary predictions and an exponential function for the duration predictions. This is intended to help to circumvent a common issue seen in previous models (e.g. Harutyunyan et al. [7], as they struggle to produce predictions over the full range of durations when the data is very skewed) i.e. it effectively allows the upstream network to model $\log(\text{LoS})$ instead of LoS. This distribution is much closer to a Gaussian distribution than the remaining LoS distribution. Batch normalisation [10] and dropout [26] is used to regularise the model. The hyperparameter search methodology is described in Section A.6.

A.2 Other Encoders

The other encoders are a two-layer *Long-Short Term Memory* network (LSTM) [9], and a Transformer encoder network [30] (see Section A.6 for their hyperparameters).

The LSTM is very similar to the one used in a recent eICU benchmark paper including LoS prediction [23]. The Transformer implementation is the same as the original encoder in Vaswani et al. [30], except that we add temporal masking to impose causality[3].

In each of the alternative encoders, the data (including decay indicators) and the non-time series components of the models were the same as in TPC (Figure 4).

[2] 'Receptive field' refers to the width of the filter. For TCNs this corresponds to a timespan.

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

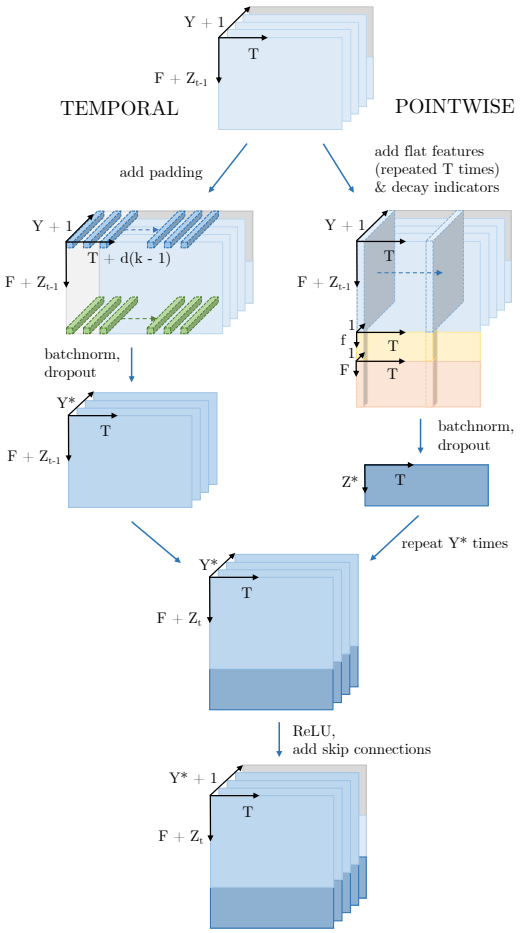

Figure 3: One layer of the TPC model. F is the number of time series features. T is the time series length. Y is the number of temporal channels per feature in the *previous* TPC layer (except for the first layer where Y is 1; decay indicators (explained under 'Time Series' in A.5) make up this channel). Z_{t-1} is the cumulative number of pointwise outputs from *all previous* TPC layers. Y^* and Z^* are the number of temporal channels per feature and pointwise outputs respectively in the *current* TPC layer. $Z_t = Z_{t-1} + Z^*$. The differently coloured temporal filters indicate independent parameters. d is the temporal dilation, k is the kernel size. Decay indicator features (A.5) are shown in orange, f static features are shown in yellow. The skip connections consist of F original features (grey) and Z_{t+1} pointwise outputs (light blue). We ignore the batch dimension for clarity.

A.3 Remaining Length of Stay and Ventilation Duration Tasks

We assign a remaining LoS and remaining VD target to each hour of the ventilation episode, ending when the patient dies or is extubated. The remaining LoS is calculated by subtracting the time elapsed in the ICU from the total LoS. The remaining VD is calculated by subtracting the time elapsed in the ventilation episode from the total ventilation duration. We only train on data within the first 21 days of the ventilation episode to protect against batches becoming overly long and slowing down training.

Loss Function The remaining LoS and VD each have a significant positive skew which makes the task more challenging. We partly circumvent this by replacing the commonly used mean squared error (MSE) loss with mean squared *log* error (MSLE). MSLE penalises *proportional* over absolute error, which seems more reasonable when considering an error of 5 days in the context of a 2-day vs. a 30-day stay. We also report MAD, as it is the preferred metric of Harutyunyan et al. [7].

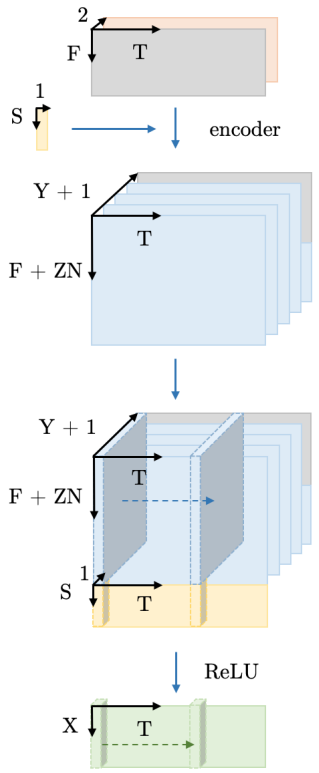

Figure 4: Overview of the encoding framework. F, T, Y^*, Z^*, Z_t and f are defined in the caption to Figure 3. The original time series (grey) along with the decay indicators (orange) (explained under 'Time Series' in Section A.5) are processed by n TPC layers. If a baseline model were used instead of TPC, the time series output dimensions would be M x T, where M is the LSTM hidden size or d_{model} in the Transformer (this is in place of the light blue and grey output in the TPC model). The diagnoses, d, are embedded by a diagnosis encoder – a single fully connected layer of size D. The time series (blue and grey), diagnosis embedding (purple) and static features (yellow) are concatenated along the feature axis, and a two-layer pointwise convolution is applied to obtain the embeddings (green).

A.4 Feature Selection

We selected time series variables, mainly from the *numericitems* table. We used a semi-automatic process for feature selection. To be included, the variable had to be present in at least 25% of patient stays, and these were further narrowed down with expert advice. We extracted diagnosissubgroups using a query from the AmsterdamUMCdb github repository [28] and ventilator settings from *listitems*, which were used to create some of the additional features listed in Table 3. We also extracted 14 static (non time-varying) features from the *admissions* table (see Table 2).

A.5 Feature Pre-processing

Static Features 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.

Time Series For each ventilation episode, 31 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. Where there is missingness, we forward-fill to bridge the

Table 2: Static features used in the model. Null Height and Null Weight were added as indicator variables to indicate when the height or weight were missing and have been imputed with the mean value.

Feature	**Type**	**Source Table**
Gender	Binary	admissions
Age Group	Discrete	admissions
Height Group	Discrete	admissions
Weight Group	Discrete	admissions
Admission Count	Discrete	
Ventilation Episode Count	Discrete	
Urgency	Binary	admissions
Previous Ward	Categorical	admissions
Specific Location in ICU	Categorical	admissions
Physician Speciality	Categorical	admissions
Weight Source	Categorical	admissions
Height Source	Categorical	admissions
Null Height	Binary	
Null Weight	Binary	

gaps in the data[4]. After forward-filling is complete, any data recorded before the ICU admission is removed. To inform the model about where the data is stale, we add a 'decay indicator' to each feature to track how long it has been since a genuine observation was recorded. The decay is calculated as 0.8^t, where t is the time since the last recording. If it is up to date, decay is 1, and if it cannot be forward-filled, decay is 0. This is similar in spirit to the masking used by Che et al. [3].

[4]This is preferable to interpolating between the data points because in realistic scenarios the clinician would only have the most recent value and its timestamp.

Table 3: Time Series features. The features which do not have a source table were calculated from the other features available in the data and were not present in the original dataset.

Feature	**Type**	**Source Table**
ABP gemiddeld	Continuous	numericitems
Ademfreq.	Continuous	numericitems
Alb.Chem (bloed)	Continuous	numericitems
Bilirubine (bloed)	Continuous	numericitems
CRP (bloed)	Continuous	numericitems
End tidal CO2 concentratie	Continuous	numericitems
Exp. tidal volume	Continuous	numericitems
Glucose (bloed)	Continuous	numericitems
Hartfrequentie	Continuous	numericitems
Ht (bloed)	Continuous	numericitems
Kalium (bloed)	Continuous	numericitems
Kreatinine (bloed)	Continuous	numericitems
Lactaat (bloed)	Continuous	numericitems
Leuco's (bloed)	Continuous	numericitems
Natrium (bloed)	Continuous	numericitems
O2 concentratie	Continuous	numericitems
P/F ratio	Continuous	
PC	Continuous	numericitems
PEEP (Set)	Continuous	numericitems
PO2 (bloed)	Continuous	numericitems
Piek druk	Continuous	numericitems
Saturatie (Monitor)	Continuous	numericitems
Temp.	Continuous	numericitems
Thrombo's (bloed)	Continuous	numericitems
TroponineT (bloed)	Continuous	numericitems
UrineCAD	Continuous	numericitems
lung compliance	Continuous	
mandatory ventilation	Binary	
pCO2 (bloed)	Continuous	numericitems
pH (bloed)	Continuous	numericitems
patient triggered	Binary	
Time in the ICU	Discrete	

A.6 Hyperparameter Search

All the encoders 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 keeping all other parameters fixed. These parameters then remained fixed for all the models which share their non-time series specific architecture. We ran 10 hyperparameter trials to optimise the remaining parameters for the TPC, LSTM, and Transformer models. The number of epochs was determined by selecting the best validation performance from a model trained over 300 epochs. All deep learning methods were implemented in PyTorch [17] using PyTorch Lightning [6] and were optimised using Adam [12].

We also optimised for the weighting between the tasks. We simply multiplied the loss for each component by a hyperparameter. The best overall learning curves were found when the task weighting coefficients were: 0.5 for the duration tasks, 1 for the binary tasks, 1 for time series reconstruction and forecasting, and 0.5 for binary feature reconstruction.

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 allow 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 (6)	Point. Channels (14)
Temp. Dropout (0.05)	Main Dropout* (0.05)
Kernel Size (3)	
Batch Normalisation* (True)	
No. TPC Layers (6)	
Non-TPC Specific	**Global Parameters**
Batch Normalisation* (True)	Batch Size (128)
Main Dropout* (0.05)	Learning Rate (0.0001)
Final FC Layer Size* (16)	Embedding Size (128)

Table 5: The LSTM model has 8 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)	Batch Normalisation* (True)	Batch Size (128)
LSTM Dropout (0.05)	Main Dropout* (0.05)	Learning Rate (0.0001)
No. LSTM Layers (2)		Embedding Size (128)

Table 6: The Transformer model has 9 hyperparameters. 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)	Batch Normalisation* (True)	Batch Size (128)
Feedforward Size (256)	Main Dropout* (0.05)	Learning Rate (0.0001)
d_{model} (16)		
Transformer Dropout (0.05)		
No. Transformer Layers (6)		

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	
Final FC Layer Size	16	64	$\log_2$
Point. Channels	4	16	$\log_2$
Temp. Channels	4	16	$\log_2$
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

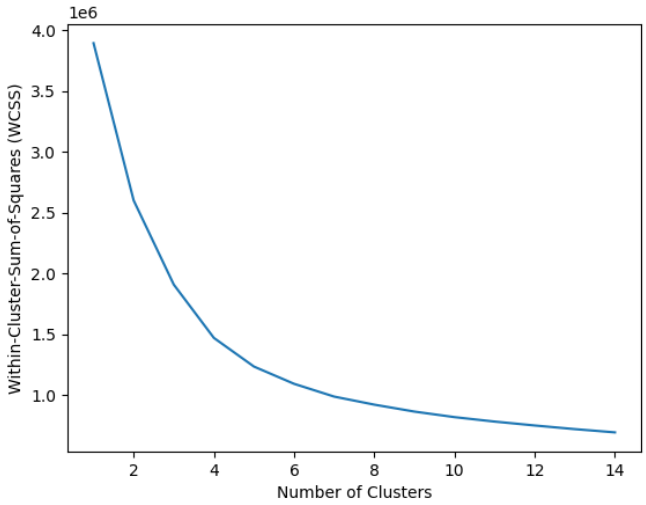

Figure 5: An example elbow plot for the TPC model.

A.7 Determining k

The value of k was determined using an average value from the elbow method across various encoders. Specifically we looked for the point at which the Within Cluster Sum of Squares (WCSS) started to tail off with increasing values of k. Figure 5 shows a typical example. We selected the value 5 across all the models.

B Additional Figures and Discussion

B.1 Performance on the Self-supervised and Unsupervised Auxillary Tasks

Interestingly, the LSTM performs the best on the unsupervised and self-supervised tasks, achieving a loss of 0.165 ± 0.003 on the reconstructive task (compared to 0.241 ± 0.003 and 0.335 ± 0.007 on the TPC and Transformer respectively). On the forecasting tasks, the pattern is similar 0.180 ± 0.003, 0.250 ± 0.003 and 0.333 ± 0.007 for the LSTM, TPC and Transformer respectively. It is unexpected that the model performs better on the reconstructive task rather than the forecasting, as it already has access to this data in the input.

B.2 Further Cluster Analysis

Below are a set of figures showing further analyses conducted on the clusters for the TPC model. We see that cluster 1 seems to contain healthy patients, who stay for a short time, are not urgent and they have a high survival rate. Clusters 2 and 4 stay for a medium length of time and sometimes receive tracheotomies, however cluster 2 is a more severe subtype with more than double the mortality. Cluster 3 represents a chronic pulmonary disease subtype who receive very long ventilation durations and stay for a long time in the hospital. Cluster 5 is a severe acute phenotype with high urgency, and they often die reasonably soon into their stay. Figure 14 gives an indication of how related the clusters are to one another. It shows that clusters 1 and 5 are the most distinct whereas clusters 2, 3 and 4 are more likely to co-exist within the same ventilation episode, as the patient's prognosis changes.

Not only do these clusters represent clinically meaningful phenotypes with differences in outcome, we see that they are repeatable with different encoders, as shown in Figure 2.

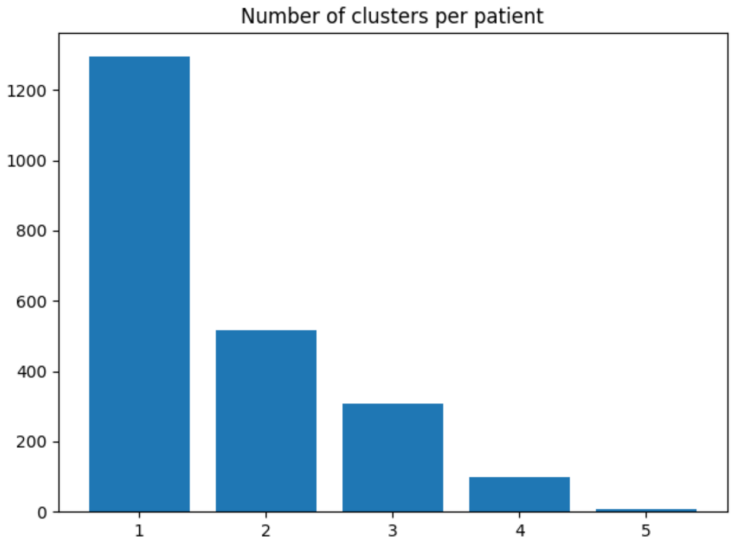

Figure 6: Number of clusters assigned per ventilation episode, shown for the TPC model. Most patients are assigned the same cluster throughout their stay, however some long stay patients can enter multiple cluster states during their episode, because events happen during the stay that alter their outlook.

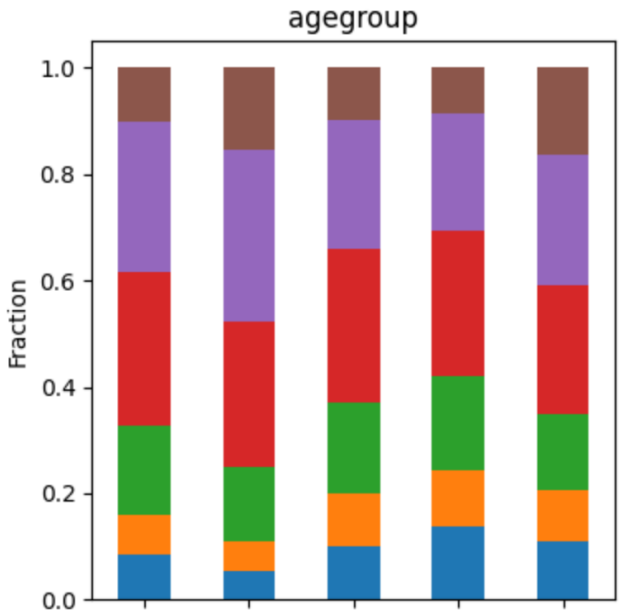

Figure 7: Difference in demographics between the clusters for the TPC model. We can see that cluster number 2 tends to have younger patients and cluster 3 and 4 is more elderly.

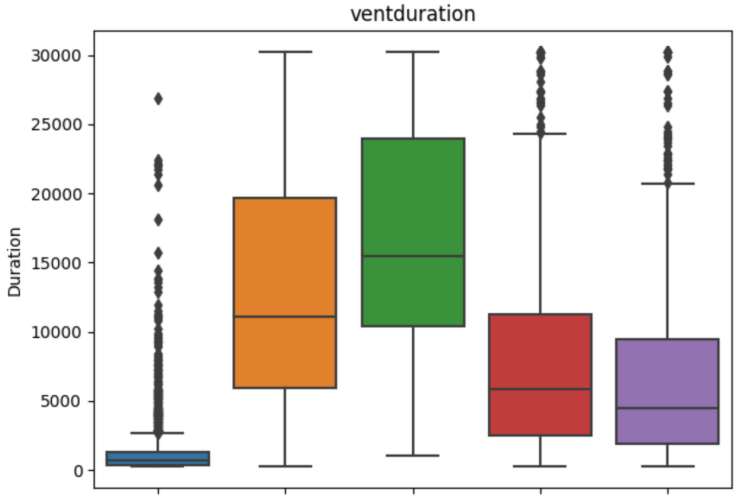

Figure 8: Cluster 3 is associated with the longest ventilation duration.

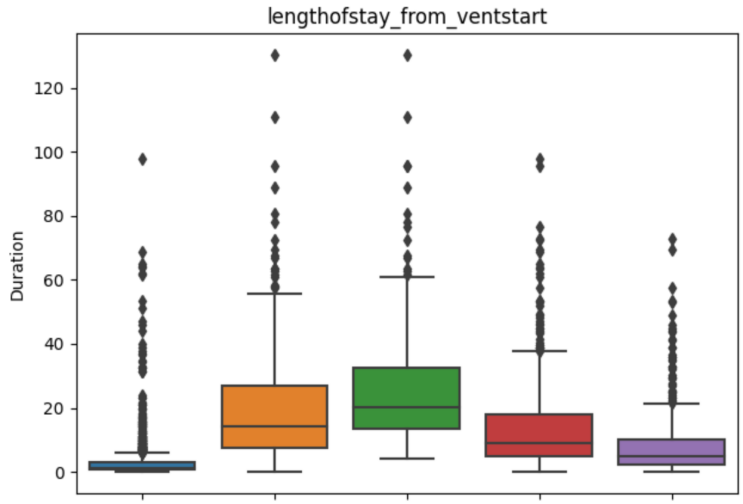

Figure 9: The length of stay distribution follows closely to the ventilation duration. The tasks are however distinct because the stay on the ICU may continue after the patient is extubated.

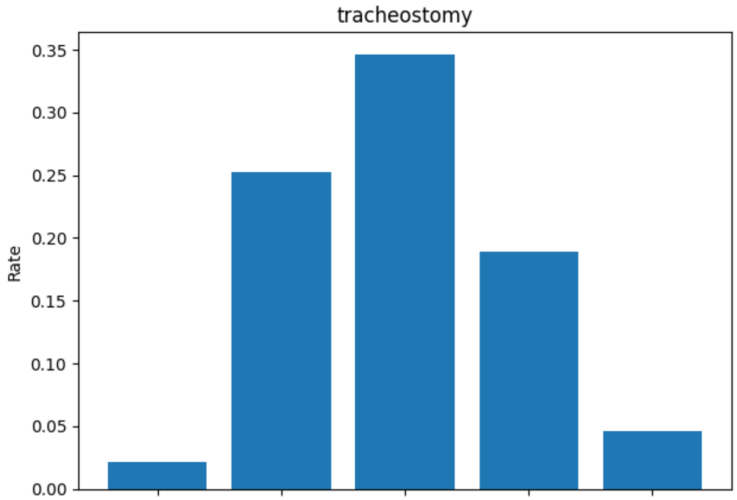

Figure 10: Cluster 3 is highly associated with tracheostomy, making it more likely to represent a long ventilation duration and chronic lung disease.

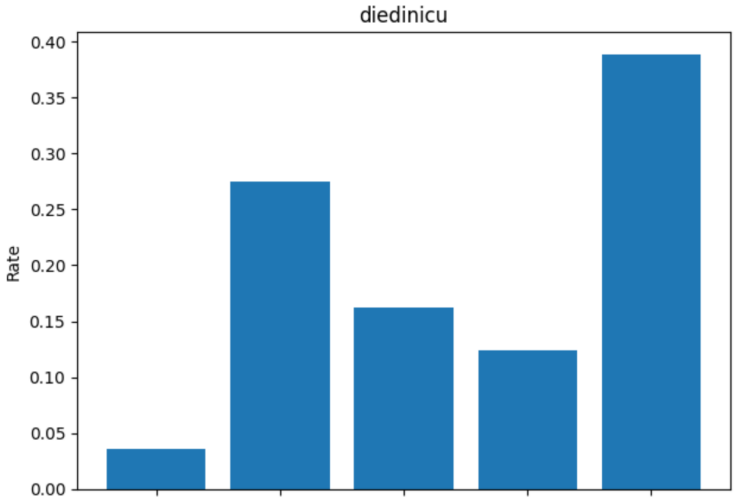

Figure 11: Mortality is most associated with cluster 2 and cluster 5. From correlating this with the other figures, it is likely that cluster 5 represents severe acute disease whereas cluster 2 is chronic lung disease.

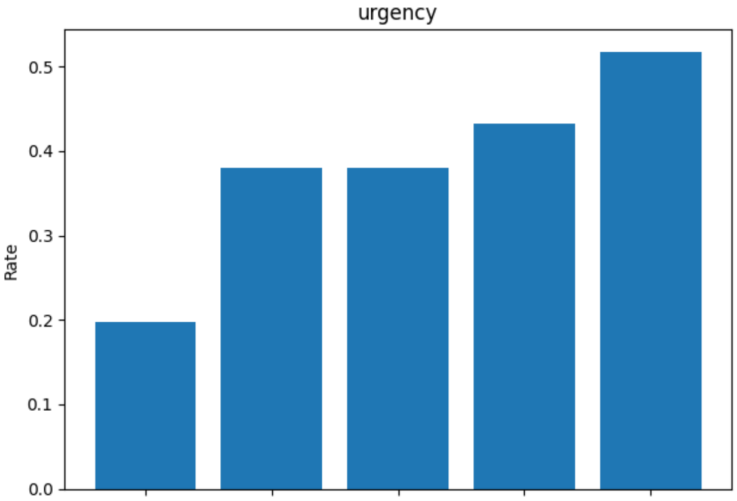

Figure 12: Urgency is most associated with cluster 5, which carries the highest mortality.

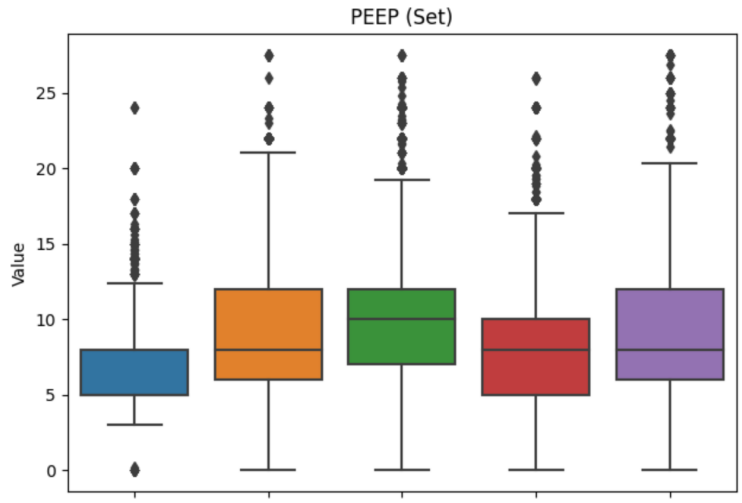

Figure 13: There are differences in the assigned treatments given to each cluster. Positive End Expiratory Pressure (PEEP) is an aspect of ventilatory treatment usually given when a patient is dealing with pulmonary oedema as it helps keep the lung spaces open. We do not see it used so much in the relatively healthy cluster 1.

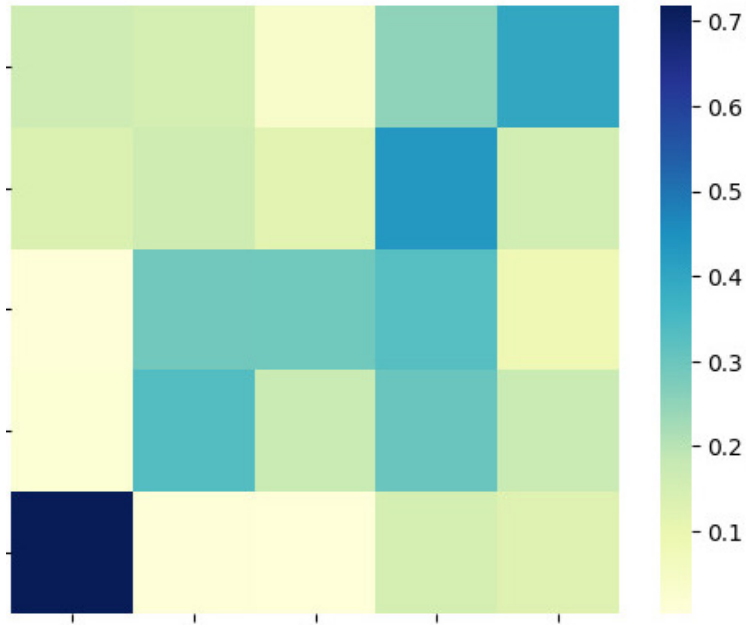

Figure 14: This schematic shows for each cluster, the percentage of time that is spent within each other cluster during the same ventilation episode. For example, cluster 1 almost always exists on its own (seen by the dark square in the bottom left). After cluster 1, cluster 5 is the next most distinct, as this is the cluster which corresponds to the acute, high mortality patients. The other clusters are more likely to co-exist in the same ventilation episode.

B.3 Contrastive Learning

Contrastive learning usually involves adding noise to the input data, such as superimposing white noise, cropping the data to remove certain values, or performing random knock out. In theory one could then obtain an embedding from the original data, and an alternative from a noisy version of the data. These embeddings should be similar because they stem from the same input data. An extra loss is added to encourage the distance between this pair of embeddings to be minimised. Not only does this encourage the distances in the embedding space to reflect true distances between data points, but it also explicitly encourages the model to remove noise from the data.