Applications of Deep Learning Methods to the MIMICIII Multi-Modal Multi-Task Clinical Inference Baseline: A Study of Transformers and Hierarchical Network Architectures

Jake Sauter, Nick Bartelo, Tom Berry, Mingxuan Zhang
Masters in Computational Biology
Weill Cornell Graduate School of Medical Sciences
New York, NY 10075

1 Introduction

Predictive medicine is a growing part of the medical field which attempts to predict the probability of the outcome of a disease for patients based on healthcare data. One of the main goals of predictive medicine is the expansion towards early disease detection and eventually disease prevention. Among all the new techniques, data driven algorithms using large amounts of patient information have the potential to provide beneficial insights. More accurate disease identifiers and signatures can be learned from data across patients and used on a single patient. As a result, the availability of healthcare data has increased enormously.

Healthcare information comes from many different modalities, such as images, text, and numerical data. These are characterized by different statistical properties, and need to be processed separately. However, through the use of multi-modal learning, it is possible to combine the results of many different data types (modalities) to create a model which is more accurate than those using one modality.

A previous work [3] introduces MIMIC-III (M3) — a dataset and benchmark for evaluating machine learning algorithms in the healthcare domain. This dataset contains multi-modal patient data collected from intensive care units such as physiological time series, clinical notes, ECG waveforms, and tabular inputs. It also defines six clinical tasks: predicting mortality, decompensation, readmission, and other outcomes, which serve as benchmarks for comparing algorithms. One recent work [7] introduced new multi-modal and multi-task models for the M3 dataset, which out-perform previous state-of-the-art results that only rely on a subset of all tasks and modalities. The numbers of the training, validation, and testing data are shown in (Supplementary Figure 1), as well as a few characteristics describing the data, such as when it is procured and the dimensionality of each modality. It is important to note that waveforms were not included in any of the models.

Specifically, this multi-modal and multi-task model received a greater aucroc and aucpr value than previous models reported by [8] and [9]. This highlights the potential of multi-task and multi-modal learning to improve the performance of algorithms in the healthcare domain. It is our goal to further increase the prediction accuracy of this model by implementing different hierarchical models, and using transformers for the time-series modality, separately. This could lead to a greater interest in healthcare systems for the use of multi-modal and multi-task machine learning models on predicting patient outcomes and many other important variables.

2 Background

2.1 Tasks

There are 6 tasks presented to this model, described by [7], outlined in (Supplemental Table 1). Of these tasks, we chose Decompensation as our target task: the goal of the task is to predict whether or not the patient will die in the next 24 hours, given all past data collected from the patient during their current ICU stay. Achieving high performance of this task could allow doctors to prioritize patients with a higher likelihood of passing away in the next 24 hours over less at-risk patients. In application 1 below, we also exploit task similarities between Decompensation and In-Hospital-Mortality, which aims at predicting whether a patient will die by the end of their stay.

Applied Machine Learning Project Report. Fall 2020.

2.2 Modalities

There are 3 modalities used in this model, described by [7] as:

Physiological Time Series We select 59 temporal physiological variables from MIMIC-III, such as diastolic blood pressure, systolic blood pressure, oxygen saturation. The full list is in Table 8. These physiological variables are recorded irregularly, and they are important indicators of the patient's condition during the ICU stay.

Clinical Notes Clinical notes are written by clinicians and nurses during the ICU stay and usually summarize topics such as reasons for admission, details of treatment, nutrition, and the patients' respiratory conditions. These clinical notes are also temporal, and they are charted sporadically.

Tabular Data For every patient, we have access to their recorded sex, age, height, and weight upon entry to the ICU, the type of the ICU, and other tabular inputs. We consider only the initial values upon entry to the ICU. Some of these fields, such as weight, may fluctuate throughout the ICU stay, and are also part of the time Series data.

2.3 Experimental Setup

Training Validation and Test data

All data used was curated in the MIMIC-III (M3) dataset, which contains multi-modal patient data collected from intensive care units such as physiological time series, clinical notes, ECG waveforms, and tabular inputs. Dimensionality of all data used are shown in Supplemental Table 2.

Performance metrics

We use the same performance metrics as used in the original model. Therefore, Decomp., In-Hospital Mortality, and Long-Term Mortality are compared using aucpr, and phenotype, length of stay, and readmission are compared using aucroc. We use the aucroc metric when there is an importance of negatives, since aucpr does not incorporate true negatives. Negatives are not important in this model for Decompensation, In-Hospital Mortality, and Long-Term Mortality due to the higher importance of detecting patient mortality, as apposed to paying more attention to patients who may have survived otherwise.

Baseline Model

All performance metrics used in our experiments were compared against those of the multi-modal and multi-task model developed by Marroquin et al [7].

3 Methods

3.1 Application 1 - Hierarchical Multi-Task Learning (HMTL)

In this application, we look to apply previous successes [6] in using Hierarchical Multi-Task Learning to exploit relationships between a primary task and auxiliary tasks. First, we remind the reader that in the case of multi-task learning, shared network parameters of each input sample lead to shared internal representations of this sample, that all tasks must find useful in order for the network to optimize its cost function. Sharing representations between related tasks has proven to allow better generalization in the context of Multi-Task learning [5]. Hierarchical multi-task learning adds a unique structure to this type of model architecture which aims to group tasks which appear to be more strongly correlated. We aim to extend the work of [7] through selective grouping of related tasks into two different hierarchical model architectures. After studying the hierarchical multi-task architectures from reference [6], shown in (Supplementary Figure 1), we developed two architectures to be implemented. The first architecture, denoted Model 1, consists of a single depth construction. The second architecture, denoted Model 2, consists of a double depth construction. We can then compare the final performance metrics of our hierarchical models to the baseline and demonstrate any improvements. Our proposed model architectures are shown in (Supplementary Figure 2).

To begin, we defined the intermediate representation dimension size of Model 1 and Model 2 as a parameter to adjust the number of weights connecting the final layers of our models. In both architectures (Supplemental Figure 2), we define the intermediate representation dimension size as the output dimension size of the the dense layer connecting the global representation to the "Long" and "Short" representations. We then further define a second intermediate representation size for the dense layer connecting the "Short" representation to the "Death Resulting" representation in Model 2, which is set to be 75% of the given dimension size, with a lower bound of 128.

We have also defined a distribution of task weights that are used in benchmarks for our experiments for both models. Using the same formulation as reference [6], we set our initial weights to sum to 1, demonstrated through the formula: $\sum_{i=1}^{t} \gamma_t = 1$ where t is the task and γ_t represents the weight of a task. Since our models are hierarchical, we assign weights to auxiliary tasks based on what tasks we believe will most likely influence the target task. For both models above, and the baseline, our target task was decompensation. Therefore,

decompensation has the largest weight. Since in-hospital mortality (ihm) predicts a very similar result to decompensation, we believe that weighing ihm greater than the other tasks will also result in a better model performance for the target task. In addition, since length of stay is also more closely related to decompensation than the remaining three tasks, this weight is given a greater value. The three remaining weights are set equal so that our formula above is satisfied. The initial task weights are shown in (Supplementary Table 3).

3.2 Application 2 - Transformers on Time Series

In this application, we explore the effect of applying Transformers to time series. Transformers are first introduced as models with encoder-decoder architectures that apply self attention mechanism to recognize sequential patterns[10]. Some recent works have been focusing on applying this model to time series data[4] and we investigate the impact of such architecture on multitask learning. In the purposed architecture, attention functions can be defined as mapping a query and some key-value pairs to the sum of values weighted by compatibility. To build an attention sub-layer, we choose to adapt the scaled dot-product attention function and project queries, keys, and values multiple times with learned linear projections[10]. The attention vectors computed from each projections are concatenated as the final attention representation. An encoder layer is formed by stacking an attention sub-layer with a fully connected feed forward network. Layer normalization operations and residual connections are applied at each sub-layers. To produce a learned embedding of time series data for the multi-task model, we stack several encoder layers and discarded the decoder architecture since no immediate predictions need to be computed.

To explore the effect of applying transformers to time series data, we adopted areas under the precision-recall curves of decompensation prediction as the performance metric. We first isolated the time series modality and compare the performance differences between computing the embedding with a LSTM and a transformer. Data pre-processing operations for the transformers were then implemented , namely positional encoding and masking, to ensure valid performances. Positional encoding provides transformers with sequence order information by adding sine and cosine waves of different frequencies to the input embedding. Time series data with positional encoding is plotted to ensure validity. Attention mask hides data from future time steps to stop information leakage. To test our attention mask, we perturbed the input data with a time threshold and confirmed that the masked transformer cannot see pass the given threshold. We also profiled the time series data and performed feature selection based on importance to promote memory and training efficiency. Lastly, we combined all modalities and observe the performances of the multitask model with our tuned transformer as the encoder for time series.

4 Experiments

4.1 Application 1 - Hierarchical Multi-Task Learning (HMTL)

4.1.1 Effects of Varying Dimension Sizes and Dropout Rate

We began our experiment with the goal of understanding the impact of the amount of parameters in the fully connected layers between the global representation and the intermediate representations. In addition, to answer if more or less dropout helps in the connection between global to intermediate representations, we perform experiments in which we vary the rate of dropout. The baseline model utilizes a dropout rate of 0.2.

We began to answer these questions by first running the baseline model, hierarchical model 1, and hierarchical model 2 with the same parameters in order to create a baseline for all models. We then varied the dimension sizes of the layers, effectively reducing the number of parameters used by the models, in an attempt to increase the generalization ability. The results demonstrated that using less global dimensions than initially formulated leads to better generalization ability. This trend will obviously cease at a minimum viable model, which we have seem to pass with dimension size 150 used in conjunction with Model 1 (Supplementary Table 5).

Knowing that diminishing the number of parameters resulted in greater performance, and still noticing worse generalization of the training data, we hypothesized that increasing the dropout rate of the lower dimensional space would result in the response of a higher aucpr. The reason for this is because increasing dropout is a common regularization technique, used to decrease over-fitting and increase generalization ability. We found that increasing the intermediate representation dropout rate to 0.4 while holding intermediate representation size constant lead to worse performance for both architectures (Supplementary Table 5).

As we saw an increase in decompensation test performance when we reduced the global dimension sizes, we next decided to further decrease them. We varied the values of the dropout rate, including values of 0 and 0.2 for the models. Our results indicate that decreasing the dropout rate from 0.2 to 0 results in smaller decompensation test performance. We have again demonstrated that a change in dropout has seemingly reduced generalization ability from the training set. The results indicate that none of the metrics received as high of a score as Model 1 did previously with a dropout rate of 0.2 and reduction in dimension size to 200 and 150. In the Table 1, we show the training validation, and testing decompensation aucpr for our experiments.

Table 1: Results for changes in dropout and dimension sizes.

Model	Dropout Rate	Dimension Size	Train Decomp Aucpr	Val Decomp Aucpr	Test Decomp Aucpr
Baseline	0.2	NA	0.237	0.374	**0.3664**
Model 1	0.2	600	0.237	0.352	0.3517
Model 1	0.4	200	0.292	0.360	0.3411
Model 1	0.2	200	0.275	0.354	**0.3666**
Model 1	0.2	150	0.258	0.364	0.3475
Model 1	0.0	150	0.357	0.363	0.3463
Model 2	0.2	600	0.235	0.357	0.3364
Model 2	0.4	200	0.266	0.354	0.3333
Model 2	0.2	200	0.287	0.344	0.3439
Model 2	0.2	150	0.278	0.359	**0.3633**
Model 2	0.0	150	0.370	0.363	0.3517

4.1.2 Primary/Auxiliary Task Loss Weight Tuning

We choose Model 1 with a dropout rate of 0.2 and dimension sizes of 200 and 150, and Model 2 with a dropout rate of 0.2 and dimension sizes 150 and 128 for the next part of our analysis, as these output the best performance metrics of all models thus far. We tested the hypothesis that due to our hierarchical structure, if we increase the decompensation task weight, thereby decreasing the in-hospital mortality auxiliary weight, there will be a value where the test aucpr reaches a peak and then begins to decline, demonstrating the importance of the interaction between decompensation and ihm. This result would demonstrate the influence of the auxiliary task, ihm, on the target task due to the hierarchical structure of the model. The reason we suspect this activity is because our hierarchical models are designed to perform most efficiently on the target task as a result of the association between grouped tasks in the architecture. Therefore, we attempt to find the best shared representation by forcing an independent shared representation between IHM and Decomp architecturally and modifying their task weights. In order to keep the scaling between auxiliary weights while calculating the new experimental weights, we find the proportion of all auxiliary task weights to the in-hospital mortality weight for the initial task weights. As we vary the decomp. task weight, we can solve for the ihm weight, and therefore all other auxiliary weights, using the derived from Supplemental Formula 1.

By using this formula, we implement weights which keep the relationship between auxiliary tasks, allowing us to make conclusions about the performance metric in regards to the change of the independent variable, i.e., the decompensation weight. The weights used for experimentation are shown in Supplemental Table 4, and the final results are shown in Supplemental Table 5.

Our results show that our hypothesis is correct. We saw a decrease in test aucpr as we both increased and decreased the decompensation task weight, which caused a corresponding decrease and increase in the in-hospital mortality auxiliary task weight.

In the case of increasing the primary task weight relative to the auxiliary tasks, the drop in performance may be due to not as useful of a representation being learned in the final layers of the model, because this representation did not have to be as useful for the auxiliary tasks. This can be likened to greedily optimizing for the primary task without considering the possibility of better optima existing when more strongly considering the importance of the auxiliary tasks. This is one of the main driving factors behind our hierarchical architectures, as more collaboration is enforced between the more strongly related tasks.

For the case of decreasing the primary task weight relative to the auxiliary tasks, the performance decrease could be attributed to too weak of a IHM usefulness constraint. When we enable complete task sharing in changing the importances of Decompensation and IHM, we see a decrease in performance, showing that these two tasks do require a different optimal representation. However, as seen before IHM does provide a meaningful constraint on the final intermediate representation shared solely between the two tasks.

4.2 Application 2 - Transformers on Time Series

4.2.1 Faulty time-series modality

Before incorporating our new encoder for the time-series modality, we explored our baseline: the model as built by Marroquin et al. The multimodal model appeared to give good performance (Supp. Fig. 3), but to be thorough, we ran the model using one modality at a time. Running the model with the time-series modality alone gave near-zero performance (0.02 AUCPR for decompensation prediction on validation data) and a flat learning curve (Supp. Fig. 3). We inspected the data processing code for this modality and found a mismatch between file headers and file contents, caused by faulty CSV parsing. Fixing the parsing resulted in the model learning properly from the time-series data (and a 17% increase in AUCPR on the multimodal model).

4.2.2 Leakage

Next, we replaced the LSTM encoder with a Transformer encoder [11] as described in Methods. Running the model with this new encoder gave excellent performance (0.43 AUCPR for decompensation prediction on validation data), but to be thorough, we investigated other metrics as well. Performance on the length of stay metric was too good to be believed: 0.999 AUCPR (Fig. 1). We theorized that label information was likely leaking into our embedding. It seemed plausible that leakage could arise from converting the LSTM to a Transformer, because while LSTMs can reference only earlier sequence data when computing a representation of each sequence point, Transformers can attend to positions both earlier and later in the sequence.

Accordingly, we implemented a mask on the sequence data that would allow the Transformer to attend to earlier sequence positions only, at each sequence point. We confirmed that this masking achieved the desired inhibition by using the Transformer to encode two vectors, which were identical in their first ten sequence points and afterwards divergent. As expected, the encoded vectors were also identical in their first ten sequence points, and afterwards diverged, demonstrating that the encoder did not attend to later sequence points when constructing each point's embedding.

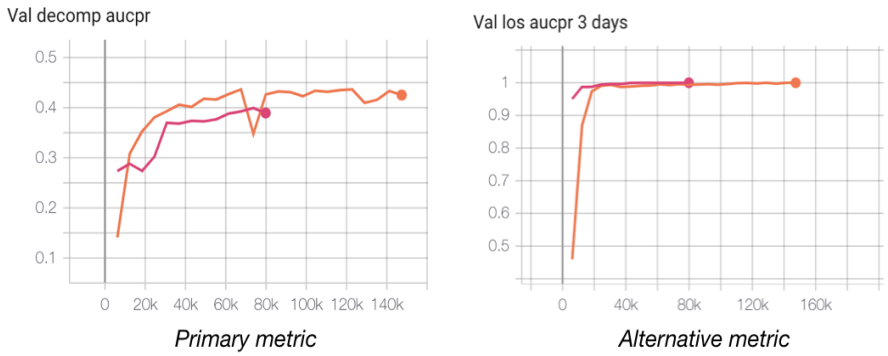


Figure 1: The extremely high performance of an alternative metric (right) revealed that labels were leaking into the Transformer embedding.

4.2.3 Overfitting

Reviewing the learning curves for our training, we noticed that after a number of epochs, train performance continued to rise while validation performance stagnated (Supp. Fig. 4) – indicative of overfitting. While we were already leveraging early stopping to avoid a decline in holdout performance, we decided to add regularization to the model to assist it in learning a generalizable manner of representing the data. Our two approaches were to add dropout and to simplify the model by reducing the number of units in hidden layers, using a variety of hyperparameter choices. Unfortunately, while these methods clearly impaired train performance, we did not observe an increase in validation performance (Supp. Fig. 4).

4.2.4 Feature investigation

Inspecting the fraction of the time that our time-series input features are present (Supp. Fig. 5), we observed that most features are present at fewer than 10% of time points. Hypothesizing that it may be difficult to learn patterns from features that are rarely present, and that removing them may effectively reduce noise in the data, we explored reducing the number of features which are considered from 60 to 6 and to 20. Using the top 20 features gave the best performance, resulting in a 15% AUCPR improvement versus using all features.

4.2.5 Positional encoding

We explored positional encoding of the data [4] to enable the transformer's self-attention mechanism to gain an understanding of position and ordering of the data. The positional encoding is a composite of sine and cosine waves at varying frequencies, which are added elementwise to the data (Supp. Fig. 6). Incorporation of positional encoding only slightly affected data behavior(Supp. Fig. 6) and did not affect model performance.

4.2.6 Performance compared to baseline

After making the adjustments mentioned, incorporating the new Transformer encoder into the multimodal model resulted in performance extremely close to that of the baseline LSTM-containing model. AUCPR for decompensation prediction on a holdout dataset was 0.41, while performance reported by Marroquin et al was 0.408. We hypothesize that this similarity is due to the masking upon the transformer encoder, which restricts it to attending to past data at each time step, similar to an LSTM. We refer deeper investigation of this parallel to future work.

5 Conclusion

5.1 Application 1 - Hierarchical Multi-Task Learning (HMTL)

In this project, we build on work done to produce the Multi-Modal Multi-Task MIMIC-III dataset [7] and benchmark to try to raise the bar on predictive medicine, through incorporating previously successful applications of Hierarchical Multi-Task Learning [6]. We show that although there may be a hierarchical relation structure between tasks, we were unable to score significantly higher than the current MIMIC-III baseline model. There are possibly better task collaborations with different hierarchical models and task relative importances that have yet to been explored in this paper.

5.2 Application 2 - Transformers on Time Series

We have implemented and tuned a transformer model to learn the underlying sequential patterns of time series data. We show that transformers with strict attention masking does not improve the performance of multi-task learning significantly. However, by selecting the most important features, we achieved slightly improved performance compare to LSTMs with more memory efficiency and more parallelism using a small transformer. At the same time, we were able to use ReLU functions exclusively which provide more stable activation and easier computation compare to sigmoid/tanh used in LSTMs. Our results demonstrated the potential of transformers to achieve more efficient information extraction from time series modalities for multi-task learning problems. Future work can further investigate on the performance outcome of transformers with loose masking and larger parameter space.

References

[1] Pennington, Jeffrey, Richard Socher, and Christopher Manning. 2017. "Stanford Glove: Global Vectors for Word Representation." Nlp.stanford.edu› Pubs › Glovenlp.stanford.edu › Pubs › Glove. https://nlp.stanford.edu/pubs/glove.pdf.

[2] Chen Qingyu , BioWordVec & BioSentVec: pre-trained embeddings for biomedical words and sentences, (2018), GitHub repository, https://github.com/ncbi-nlp/BioSentVec

[3] Johnson, Alistair E. W., Tom J. Pollard, Lu Shen, Li-Wei H. Lehman, Mengling Feng, Mohammad Ghassemi, Benjamin Moody, Peter Szolovits, Leo Anthony Celi, and Roger G. Mark. 2016. "MIMIC-III, a Freely Accessible Critical Care Database." Scientific Data 3 (1): 160035.

[4] Wu, Neo, Bradley Green, Xue Ben, and Shawn O'Banion. 2020. "Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case." arXiv [cs.LG]. arXiv. http://arxiv.org/abs/2001.08317.

[5] Ruder, Sebastian. "An Overview of Multi-Task Learning for Deep Learning." Sebastian Ruder, Sebastian Ruder, 24 Oct. 2018, ruder.io/multi-task/.

[6] Bharadhwaj, Homanga. "Hierarchical Multi-Task Learning for Healthy Drink Classification." International Joint Conference on Neural Networks (IJCNN) (2019). https://ubiquitous.comp.nus.edu.sg/wp-content/uploads/2019/04/ijcnn2019-healthy-drink-cnn.pdf

[7] Marroquin, Edgar, et al. "A Multi-Modal and Multitask Benchmark in the Clinical Domain." International Conference on Learning Representations (ICLR) (2021). https://openreview.net/pdf?id=1MJPtHogkwX

[8] 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(1):96, 2019. ISSN 2052-4463. doi: 10.1038/s41597-019-0103-9. URL https://doi.org/10.1038/ s41597-019-0103-9.

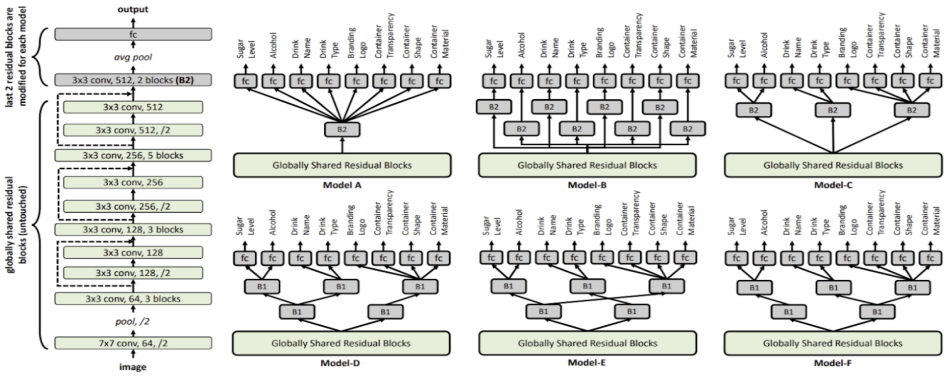
[9] Swaraj Khadanga, Karan Aggarwal, Shafiq Joty, and Jaideep Srivastava. "Using clinical notes with time series data for icu management." 2020.

[10] Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." arXiv [cs.CL]. arXiv. http://arxiv.org/abs/1706.03762.

[11] https://pytorch.org/docs/stable/generated/torch.nn.TransformerEncoder.html

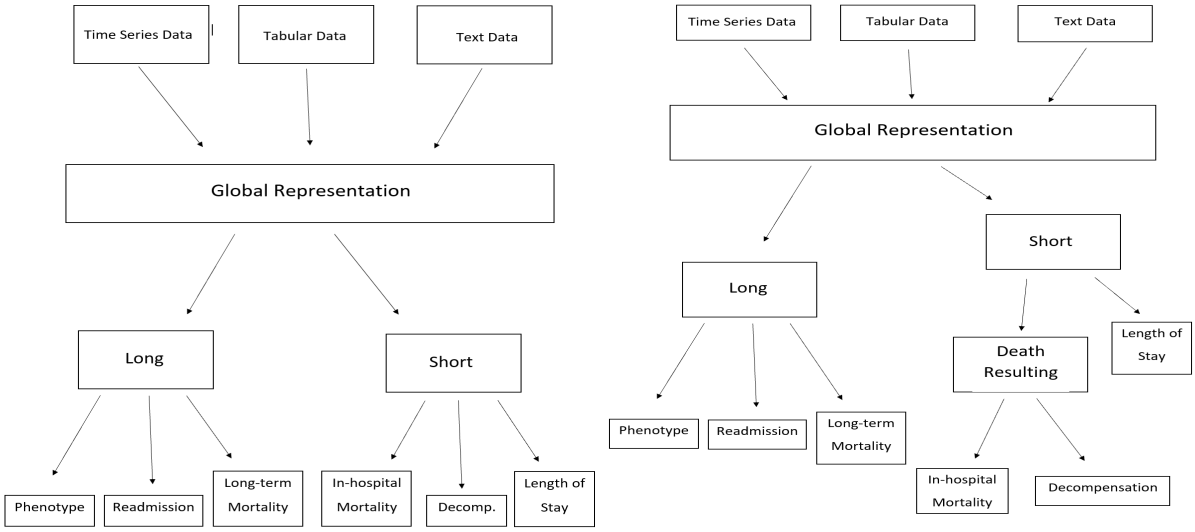
Supplemental Information

Supplemental Figure 1



Architectures used in [6].

Supplemental Figure 2

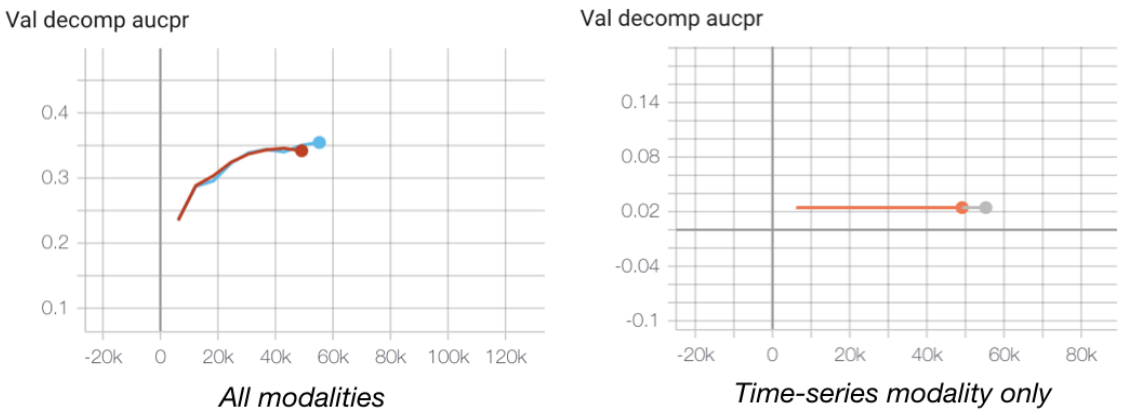


Single Depth Hierarchical Model Architecture

Double Depth Hierarchical Model Architecture

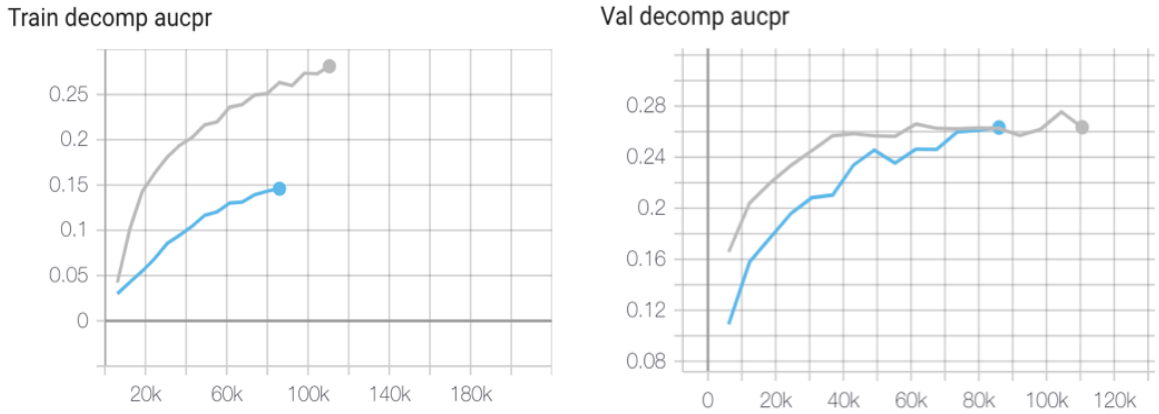
Single (Model 1) and Double (Model 2) Hierarchical Depth Model Architectures.

Supplemental Figure 3



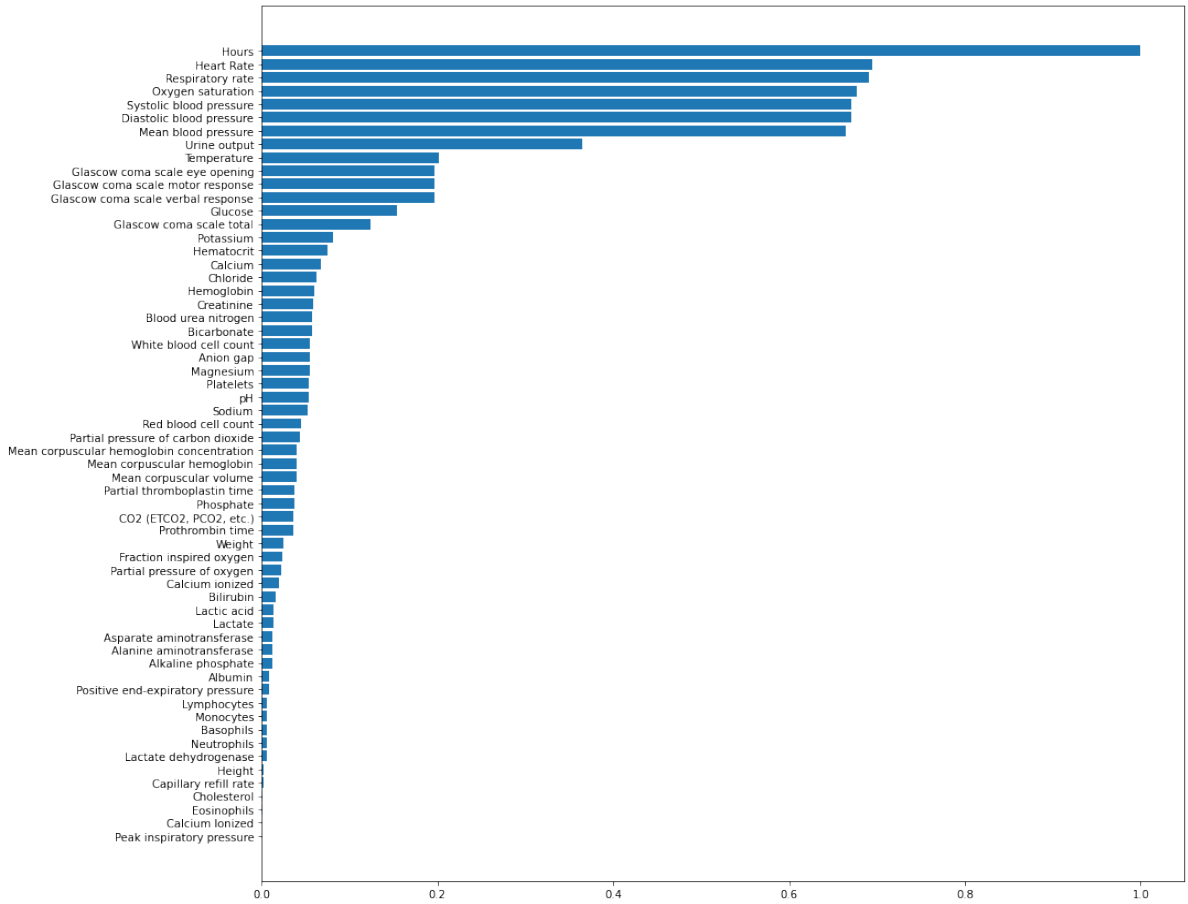
The multimodal model appeared to run properly (left) but running the time-series modality alone (right) revealed that it was not contributing to the model.

Supplemental Figure 4



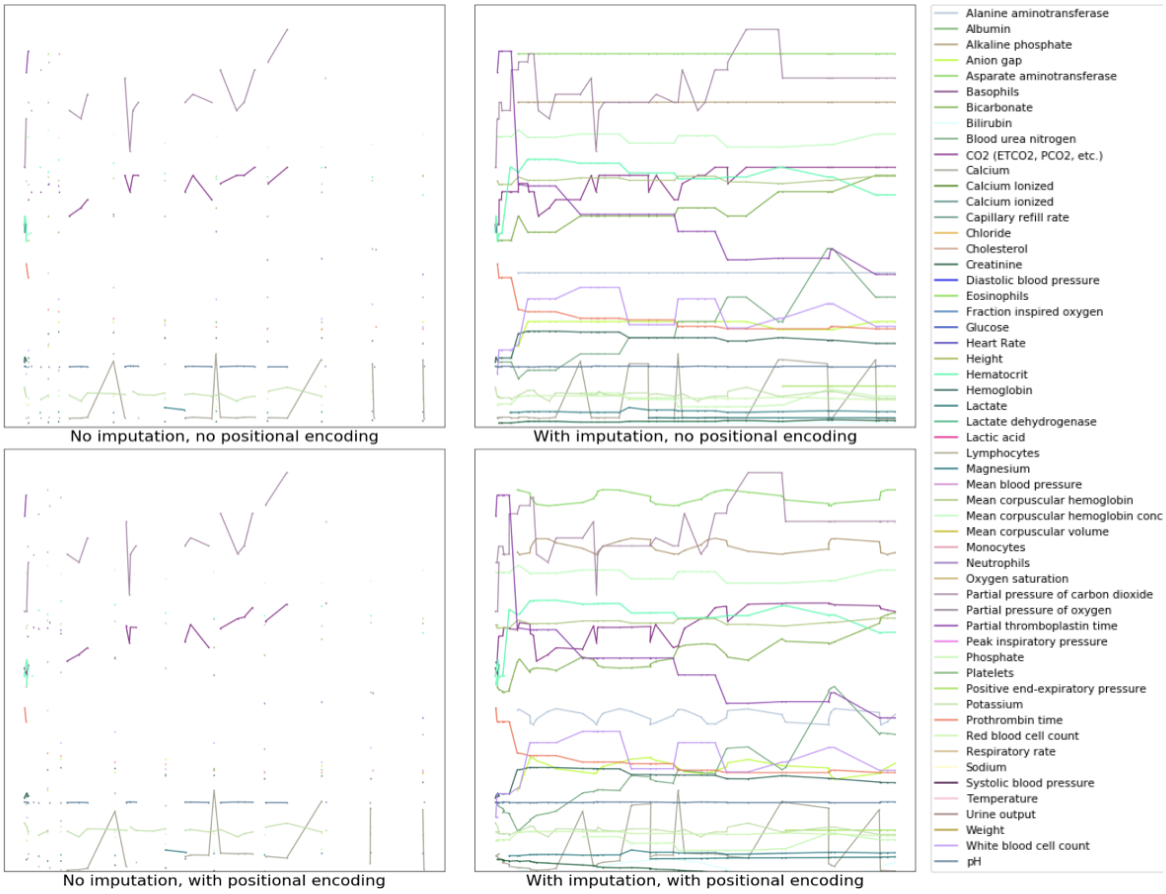
Overfitting of the data (see gray) inspired addition of regularization, which did not improve validation performance (see blue).

Supplemental Figure 5



Most features are rarely present in the data.

Supplemental Figure 6



Visualizing the time-series component of one ICU stay, with and without imputation of missing data points, and with and without incorporation of positional encoding.

Supplemental Table 1

Decompensation.	Starting from the fifth hour of the stay, a prediction is made at every hour about whether the patient will die within the next 24 hours given all the data collected to that point. Unlike in the IHM task, predictions are made on an hourly basis rather than after a set amount of time and concern the next 24 hours rather than the entire stay. As such, this task may better reflect the changing landscape of available patient information.
Length of Stay.	We predict the total duration time from admission to discharge. This task could provide useful information for medical resources allocation and scheduling. We formulate this task as a multiclass classification problem with three classes/bins (0-3 days, 3-7 days, and longer than 7 days) using only data from the 24 hours of the stay.
In-Hospital Mortality.	We observe the first 48 hours of a patient's data, and then predict whether the patient will die by the end of their stay. Mortality is one of the major concerns for any ICU unit, with limiting mortality being an ultimate goal for most ICUs.
Phenotyping.	A prediction of the patient's phenotype is made at discharge time. This is a multilabel classification task. The target label is derived from the billing code at a patients discharge, which we then convert to our 25 labels following the procedure from Harutyunyan et al. (2019).
Readmission.	We predict if another ICU stay will occur to the same patient after the discharge time of an ICU stay. Predicting readmission is useful to identify higher risk patients and minimize the waste of financial resources. We define this as a multi-class classification problem with 5 classes — readmission within 7, 7-30, 30-90, 90-365, and 365+ days or no readmission.
Long-Term Mortality.	For each ICU stay, we predict if the patient survives for more than 1 year after discharge. We frame this as a binary classification problem. Predicting long-term mortality is useful for assessing patients' well-being after discharge.

Supplemental Table 2

Task	Training Instances			Prediction time(s)	Type	Main metric
	$Train$	Val	$Test$			
Decomp.	2495.8k	360.3k	523.1k	every hour	binary	aucpr
Length of Stay	25.7k	3673	5280	24 hour	multiclass	aucroc ovr
Mortality (IH)	15.4k	2209	3234	48 hour	binary	aucpr
Phenotyping	30.7k	4383	6278	discharge time	multilabel	macro aucroc
Readmission	27.4k	3894	5659	discharge time	multiclass	aucroc ovr
Mortality (LT)	27.4k	3894	5659	discharge time	binary	aucpr

Modality	ICU stays			Dimensionality
	$Train$	Val	$Test$	
time series	30701	4383	6278	59
clinical notes	30507	4368	6247	-
tabular inputs	30701	4383	6278	6
waveforms	5567	686	1060	1

Table from [7]

Supplemental Table 3

Initial task weights.

Decomp	In-Hospital Mortality	Length of Stay	Phenotype	Long-Term Mortality	Readmission
0.25	0.2	0.15	0.133	0.133	0.133

Supplemental Table 4

Task weights used for task loss weight tuning.

Weight Setting	Decomp	In-Hospital Mortality	Length of Stay	Phenotype	Long-Term Mortality	Readmission
Initial	0.25	0.2	0.15	0.133	0.133	0.133
Increase Decomp	0.30	0.187	0.14	0.124	0.124	0.124
Decrease Decomp	0.225	0.225	0.169	0.150	0.150	0.150

Supplemental Table 5

Table 3: Task weights used for task loss weight tuning.

Model	Weight Setting	Train Decomp Aucpr	Val Decomp Aucpr	Test Decomp Aucpr
Baseline	Increase Decomp	0.210	0.374	0.3701
Baseline	Initial	0.237	0.374	0.3664
Baseline	Decrease Decomp	0.281	0.377	0.3446
Optimal Model 1	Increase Decomp	0.331	0.357	0.359
Optimal Model 1	Initial	0.275	0.354	0.3666
Optimal Model 1	Decrease Decomp	0.272	0.351	0.356
Optimal Model 2	Increase Decomp	0.292	0.357	0.3489
Optimal Model 2	Initial	0.278	0.359	0.3633
Optimal Model 2	Decrease Decomp	0.290	0.354	0.3478

Supplemental Formula 1

$$\gamma_{ihm} + \gamma_{los} + \gamma_{pheno} + \gamma_{ltm} + \gamma_{readmission} = 1 - \gamma_{decomp}$$

By construction: $\gamma_{pheno} = \gamma_{ltm} = \gamma_{readmission} \rightarrow \gamma_{ihm} + \gamma_{los} + 3 * \gamma_{pheno} = 1 - \gamma_{decomp}$

Proportions between initial auxiliary weights: $\gamma_{los} = 0.75 * \gamma_{ihm}$ and $\gamma_{pheno} = \frac{2}{3} * \gamma_{ihm}$

Plugging in the relationships between auxiliary weights and solving for γ_{ihm}: $\gamma_{ihm} = \frac{1-\gamma_{decomp}}{3.75}$