# Predicting Pediatric Patient Discharge Times
## CS230 Project Milestone

Andrew Ward
Stanford University
450 Serra Mall
atward@stanford.edu

## Abstract

*This report addresses the problem of predicting hospital patient discharge times using several machine learning and deep learning models. I consider over 1,000 pediatric patients from Lucile Packard Children's Hospital in this study, and attempt to predict whether a patient will be discharged the next day given a summary of patient characteristics for the current day. Using the AUROC metric, I present a comparison of a random forest model, a fully connected network, and an LSTM-based model. I perform a feature ablation study to gain insights into model operation. Code available at [1].*

## 1. Introduction

This project focuses on using flowsheet information inputted by nurses, as well as patient characteristics, to provide an accurate prediction of when a patient will be discharged (leave the hospital). Accurate discharge predictions are very important in facilitating smooth hospital operations. Specifically, knowing how many patients will be leaving a unit on a particular day can be used to forecast how many beds will be available in that unit. This, in turn, can be matched with the number of patients expected to arrive (which is mostly known, as a majority of hospital patients arrive because of electively scheduled surgeries), and can then be used to determine if patients need to be moved in order to make room for new intensive care patients. Additionally, having an accurate count of the number of patients in a unit can assist with nurse staffing decisions.

When considering the patient discharge prediction problem, in general, the earlier a prediction needs to be made in a patient's stay, the more difficult the problem becomes. Fortunately, the most operationally useful metric is to consider whether a patient will be discharged the next day. A 24-hour notice of patient discharge gives the hospital enough time to make decisions with the information, and is a very tractable prediction problem.

In this paper, I explore the possibility of using deep learning techniques to predict hospital discharges one day in advance. The dataset used in this study consists of 1,081 pediatric patients who stayed in the Cardiovascular Intensive Care Unit at Lucile Packard Children's Hospital (LPCH). For these patients, I have access to their characteristics (such as gender and age) and all of their Flowsheet information, which contains all of the data that nurses inputted into the Electronic Medical Record (EMR) at the patient's bedside. This data is not comprehensive, and requires substantial preprocessing to be used in a deep learning system. However, if the results of this study are promising, the models, techniques, and historical dataset used here can be expanded to create a predictive tool that can be used in real-time at LPCH to assist with operational decisions.

## 2. Related Work

As patient discharge planning is a universal problem across all hospitals, many previous studies have attempted to build predictive systems concerning patient discharges. Often, in adult patients, the location of discharge carries as much, if not more, importance than the date of discharge. In particular, three studies have assessed the predictive factors regarding whether patients who have suffered a stroke will be discharged to institutional care facilities [2] [3] [4].

For pediatric patients, discharge times, rather than discharge locations, are more studied. Similar to this study, Temple et al. examined statistical correlations to predict whether a patient will be discharged in the next 2-10 days from the Neonatal Intensive Care Unit (NICU) [5] . This study used random forest models on 26 features and achieved an AUROC of 0.865 when predicting discharge within two days.

Deep learning has also been used in discharge predictions. Yang et al. used deep learning to predict which medicines would be prescribed to a patient by considering only the patient data available at admission [6]. Another relevant study which does not concern discharge predictions was done by Avati et al. [7]. This study used a similar input

| Field | Example Value |
|---|---|
| Blood Pressure | $120/80$ |
| IVIG Volume | 1324 mL |
| Department Name | CVICU |
| Oxygen Level | 0.95 |
| Cough | Strong |
| % Weight Change | 127% |

Table 1: Examples of typical flowsheet values in the dataset.

dataset consisting of EMR patient data to generate an accurate prediction of patient mortality to be used in palliative care. One other promising recent study used deep learning based on EMR data for predictions [8]. This study attempted to predict the diagnosis and medications that would be prescribed to a patient on that patient's next visit to the hospital, given the EMR data from that patient's prior visit.

All of these prior studies indicate that both clinically and operationally relevant predictions can be made using a deep learning framework which takes EMR data as inputs, which provides a promising motivation for this study.

## 3. Methods

### 3.1. Dataset and Preprocessing

The patient cohort in this study consists of de-identified records of all of the pediatric patients that visited the cardiovascular intensive care unit (CVICU) at the Lucile Packard Children's Hospital from January 1, 2015 to October 1, 2016. In total, there are $1,081$ patient-visits considered in the study. In the rest of the study, I refer to a patient-visit simply as a patient.

For these patients, all of the flowsheet data from the electronic medical record (EMR) is available. The flowsheet data consists of numerous values that are inputted by a patient's attending nurse at the bedside; some example flowsheet values are shown in Table 1.

Each row in this dataset corresponds to a single data entry of a single field of a patient flowsheet. More specifically, one row consists of: 1) a de-identified patient number internal to LPCH, 2) a timestamp, 3) a field name, e.g. "BLOOD_PRESSURE", and 4) a value, e.g. $120/80$. These data are further complicated by their input time irregularities. As mentioned, data are only recorded when a nurse is at the patient's bedside (there are a few exceptions to this; for example, when connected to a respirator, a patient's oxygen level is automatically recorded). This results in very few data fields being inputted at regular intervals. Furthermore, a nurse will only record data that he or she thinks is relevant to the patient (and data that are required by hospital protocols). In fact, a majority of the fields (e.g. IVIG medication volume) are only recorded when certain patient conditions exist (e.g. the patient has an intravenous line).

This data format presents a significant challenge. There are over $3,700$ fields that can be recorded by a nurse, but a vast majority of these fields are very rarely recorded (less than 1 record per 100,000 records). Some of the data fields need preprocessing (e.g. blood pressure's string value of $120/80$ needs to be split up into two or three distinct numeric values), some fields are drop-down menus where multiple responses can be checked, and some of these data are text fields which, in addition to appearing very rarely, are difficult for a machine to interpret.

My strategy in preprocessing this data is as follows: I collapse all of the data for each patient, for each day, into a single data point (which I call a "patient-day"). This patient-day includes a summary of all of the information that was recorded about a particular patient within a 24 hour time slot (if the patient was admitted or discharged within the previous 24 hours, this patient-day will include a summary only accounting for the time the patient was in the hospital).

Since neural networks require the features to have the same dimension for every data point, if I include a feature for one patient, I have to include it for all patient. Therefore, I only include the fields (of which there are $3,700+$) that are most commonly recorded in the dataset. This results in $48$ numeric features (after preprocessing of fields like blood pressure), and $123$ categorical/text features.

For the selected numeric features, I compute an average, standard deviation, min, and max value based on all the measurements taken for that patient on that day. For text fields with fewer than 50 unique values, I generate one-hot binary vectors. For unique text fields (e.g. "NURSE_COMMENTS"), I create a binary vector indicating if the field is filled or not. This clearly loses information; in the future, it could be interesting to generate a text score using an average word2vec [9] or GloVe [10] score for the words included in the comments.

Even only looking at the 170 values that are most commonly recorded, the resulting feature vectors are 88% empty. To ameliorate this, I use the following simple imputation [11]: if the value is missing for a particular patient-day, I fill it with the mean value across the dataset. I also include a binary "is-NA" column for each variable.

### 3.2. Outcome Variable and Data Splits

The discharge times for each patient are accurately recorded. This clean and well-established outcome was one of the main motivators of this study. I create a binary outcome variable for each patient-day, using the patient's discharge time, that indicates if the patient will be discharged in the next 24 hours. The dataset consists of $18,471$ patient-days (from the original $1,081$ patients). The length of stays of these patients is very skewed toward shorter stays; the

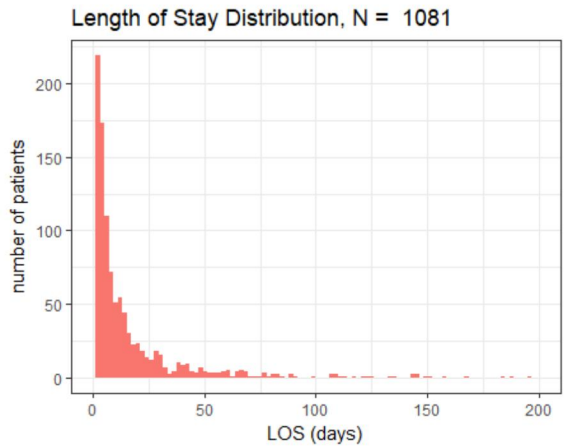


Figure 1: Distribution of patient length of stays, in days

| Split | Percent | # Patients | # Patient-Days |
|---|---|---|---|
| Training | 0.80 | 864 | 14,605 |
| Validation | 0.10 | 107 | 1,827 |
| Test | 0.10 | 110 | 2,039 |

Table 2: Training, Validation, and Test splits of the data, showing the percentage of each split, and the number of patients and patient-days included in each split.

distribution of patient length of stays, in days, is shown in Figure 1.

Although more than half of the patients remain in the hospital for less than 7 days, the distribution has a very long tail, with one patient remaining in the hospital for 590 days. I account for this when making the training, validation, and test set splits.

Since deep learning systems have been proven to achieve higher performance given more training data, I decided to use an 80-10-10 training-val-test set split, which I divided based on patients. The breakdown of the splits is shown in Table 2. I wanted the patient distributions (with respect to length of stay (LOS)) to be similar in all three data splits, so I first created three patient subsets: patients with LOS less than 14 days, patients with LOS between 14 and 60 days, and patients with LOS more than 60 days. I then selected a training, validation, and test split from each of these three subsets.

### 3.3. Models

To solve the binary discharge prediction task, I implemented three different models. The first model used random forests [12], the second model used a fully connected neural network, and the third model used an LSTM [13].

I implemented the random forest model in R, and used the validation set to tune the feature selection parameter $p$. For the final model, I used $p = \#\text{features}/4$. For both training and testing, I treated each patient-day as an independent sample, and fed it in to the random forest model and got a single binary output.

For the fully connected neural net, I used PyTorch, and modified the existing "vision" code provided by the teaching staff [14] to create a 4-layer network, with dropout [15] and batchnorm [16] added for regularization and stability. The first hidden layer has 300 nodes and a dropout rate of 0.4, the second layer has 300 nodes and a dropout rate of 0.3, and the third hidden layer has 100 nodes and a dropout rate of 0.2. I use the Adam optimizer [17] with a learning rate of $10^{-4}$, and binary cross-entropy loss (after using a Sigmoid activation on the output layer). Like the random forest model, I treated each patient-day as an independent data sample for this model.

The LSTM model also builds off of a combination of the teaching staff code [14] and the PyTorch tutorials [18]. The architecture consists of the inputs being fed into an LSTM, and then the LSTM outputs being fed into a 2-layer fully connected net. The LSTM has hidden dimension 500 with 2 layers and a dropout rate of 0.2; the first FC layer has 100 nodes and a dropout rate of 0.3, and the second FC layer has 10 nodes with a dropout rate of 0.1.

Unlike the FC-Net and the random forest model, for the LSTM I considered the patient-day samples in aggregate, grouped by the patient. I used a many-to-many architecture, so that each input resulted in an output prediction. In this way, I was able to get the same number of outputs as both the FC-Net and the random forest. However, I fed the patient-days into the model sequentially, so the LSTM was able to also take advantage of the temporal structure of the data (ideally, for each patient, the LSTM would output a sequence of zeros, followed by a single "1" on the last day the patient was in the hospital).

### 3.4. Metrics

With all of the preprocessing, each of the 18,471 patient-days consisted of 855 numeric features. Each patient has exactly one associated positive binary label, since each patient was discharged exactly once (recall that a "patient" in this study is a de-identified number associated with one patient's singular visit to the LPCH). So, there are 1,081 positive samples and 17,390 negative samples in this dataset.

Since this is such a skewed distribution, a simple accuracy metric is insufficient. We instead look at the AUROC score, which is a measure of the area under the Receiver Operating Characteristic (ROC) curve. The ROC is a plot of true positive rate (also called sensitivity or recall) plotted against false positive rate ($1-$specificity, or $1-$true negative rate). In effect, the ROC shows the tradeoff between true positive rate and false positive rate for a given model; ideally, a model achieves $100\%$ true positive rate and $0\%$

| Model | % AUROC score | | |
|---|---|---|---|
| | Train | Val | Test |
| Random Forest | 0.979 | 0.867 | 0.897 |
| FC-Net | 0.811 | 0.771 | 0.713 |
| LSTM | 0.761 | 0.740 | 0.736 |

Table 3: Results of the final versions of the three models, reported on the AUROC score. The random forest model outperformed both the fully connected network and the LSTM in predicting whether patients will be discharged within 24 hours.

false positive rate, which corresponds to the upper left point of the ROC plot. However, this is rarely obtained in practice, but the area under this curve, the AUROC score, gives a good sense for how well the model is performing.

In practice, any specific point on an ROC curve can be realized by a model. This operating point can be chosen once the model is trained according to the needs and desires of the hospital.

## 4. Results

Table 3 shows the results of evaluating the best model for the random forest, the fully connected net, and the LSTM model on the training, validation, and test datasets. The FC model performed substantially better than either deep learning models. The ROC plot for the random forest is shown in Figure 2. I followed the procedure outlined in [14] when choosing the best model, i.e. the best model was chosen to be the one that performed best on the validation dataset.

In Figure 2 and in Table 3, we see that the random forest performs substantially better on the training set than on the validation and test sets. This indicates that, even with tuning the parameter $p$, the model still overfits to the training data. However, even with this overfitting, the model achieves an AUROC of 0.897 on the test set, and can be implemeted with, for example, a true positive rate of $70\%$ and a false positive rate of $10\%$ (where I assume that, operationally, we want few false positives).

The ROC curve for the FC-Net model is shown in Figure 3, and the ROC curve for the LSTM model is shown in Figure 4. These two models performed similarly, with the FC-Net having a slightly higher AUROC on the training and validation sets, and the LSTM having a slightly higher score on the test set. The fact that the FC-Net has a significantly lower score on the test set than on the validation set could indicate that the model (based on the extensive hyperparameter tuning) has overfit to the validation set. In the future, more data should be collected to address this overfitting.

To gain more insight into the FC-model, I performed a feature ablation study in a manner similar to [7]. In this

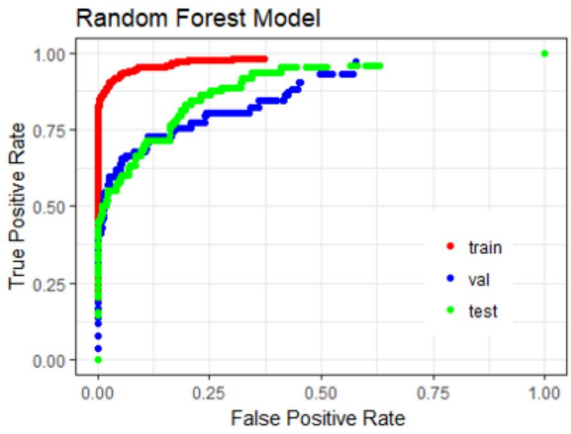


Figure 2: ROC curves for the random forest model on the training, validation, and test sets.

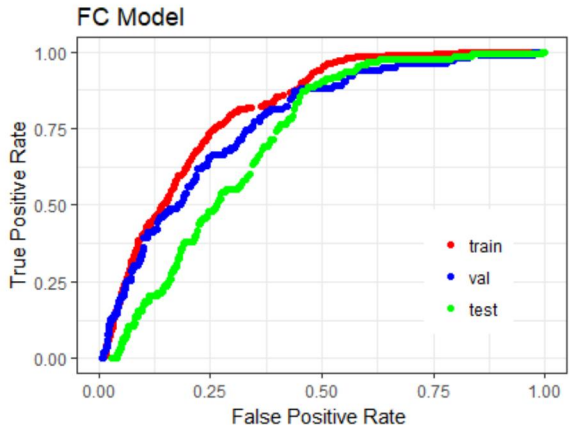


Figure 3: ROC curves for the FC-Net model on the training, validation, and test sets.

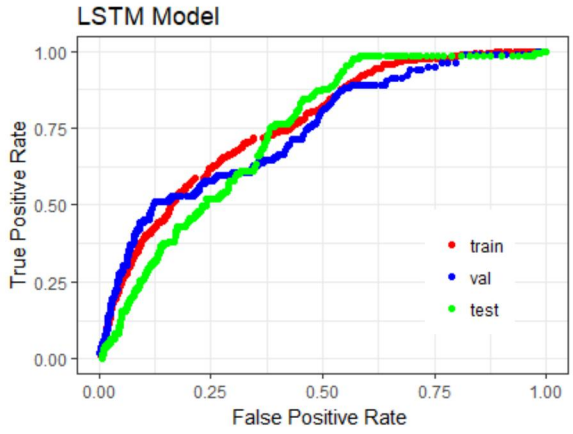


Figure 4: ROC curves for the LSTM model on the training, validation, and test sets.

| Feature | % Change in AUROC | | |
|---|---|---|---|
| | Train | Val | Test |
| Pulse Oxymetry | -6.269 | -4.846 | -6.609 |
| PEWS $O_2$ Saturation | -2.005 | -0.367 | -0.116 |
| Skin Braden Scale | -0.945 | -0.084 | 1.285 |
| Peripheral Pulse | -0.724 | -2.092 | 0.629 |
| $FIO_2$ | -0.489 | 0.246 | -0.612 |
| OR Oxygenation | -0.235 | -0.247 | 0.041 |
| Treprostinil Vol. | -0.172 | 0.388 | 0.042 |
| IVIG Vol. | -0.167 | 0.393 | 0.215 |
| Admit Type | -0.15 | -0.176 | -0.295 |
| Insulin Vol. | -0.109 | 0.127 | 1.672 |
| Core Body Temp. | 0.066 | 0.081 | 0.104 |

Table 4: Ablation study results. Each row shows the percent change in AUROC (for training, validation, and test sets) when that feature is masked and the data is passed through the fully connected model (which was trained beforehand, with no features masked). Percentage change to the AUROC score are calculated relative to the maximum AUROC score, i.e. 1.

analysis, I used the fully trained FC-Net model, and masked features one at a time. I masked numeric and categorical features differently. If the feature was numeric, I set the average, standard deviation, min, and max values to the average values in the dataset, and I set the "is-NA" variable associated with this feature to "1". If the feature was categorical, I set all of the one-hot vector values associated with that feature to "0", and I set the "is-NA" variable to "1".

Table 4 shows the results of the ablation study. I chose the 11 features for which the ablation of that feature caused the largest absolute value difference in the AUROC score. This analysis shows that Pulse Oxymetry is a very important variable; when it was ablated, the AUROC scores dropped $\sim 5 - 7\%$ on all of the data sets.

Some interesting behavior is also apparent when examining Table 4 in more detail. For example, the removal of "core body temperature" actually caused the model to perform better on all three subsets of the data; this could indicate that the preprocessing step, in which averages were inputted for all missing values, actually provided more confusion than help. We also see that for other features, the model performed worse on the training set but better on the validation and/or test set without the feature. However, since the average feature only caused a change on the order of $\sim 0.001\%$, the large changes (either positive or negative) indicate that the model was relying on these features to make predictions.

# 5. Conclusions and Future Work

In this study, I have used machine learning and deep learning techniques to build a predictive model that determines whether a patient will be discharged within 24 hours. I used a dataset comprised of $1,081$ patients from the CVICU at Lucile Packard Children's Hospital, who stayed a combined total of $18,471$ days in the hospital. I extracted $171$ features from the flowsheet dataset based on the frequency of the recorded features, and used a random forest model, a fully connected neural network, and an LSTM sequence model to predict a binary outcome variable. I used the AUROC metric to evaluate these models on the prediction task, presented ROC curves for the three models, and done a feature ablation study to gain more insight in to the workings of the FC model. The random forest model outperformed both deep learning models; this is likely due to the data preprocessing and the dataset size.

## 5.1. Alternate feature representation

It is possible that a large reason the random forest performed so well was because the data imputation was done in a way that was suited to random forest model usage [11]. With the addition of the "is-NA" feature, a decision tree model could simply have a branch that looked at the "is-NA" variable for a specific feature, and if that variable was "1", the tree would no longer consider that feature.

One possible solution would be to have an LSTM consider the raw flowsheet inputs, which consist of a time, a field name, and a value. The field names are inputted sequentially like a nurse, and, like words, the order in which the fields are inputted is highly correlated. Some additional modeling would have to take in to account how to handle the actual values, but an LSTM-based model operating on the raw data would be a promising avenue of future work.

## 5.2. Additional Dataset

A limiting factor in the deep learning analysis that can be done on this problem is the size of the dataset. Aside from the small number of patients, when the data is broken up into patient-days, patients with longer length of stays are disproportionally represented in the dataset (the patient who stayed in the hospital for 590 days would contribute 590 times as much to the loss function as a patient only staying one day). The publicly available MIMIC III Dataset [19] provides an attractive alternative to this small dataset. This collection has records of 58,976 patients, of which 8,155 are pediatric patients. The data format is very similar, but the naming is very different than Lucile Packard data, with very little direct overlap. So a transfer learning approach could be difficult, but it would be interesting to study how the efficacy of the deep learning models improves with more patients.

## References


[1] A. Ward, "cs230-project." https://github.com/atward424/cs230-project, 2017.

[2] J. K. Burton, E. E. C. Ferguson, A. J. Barugh, K. E. Walesby, A. M. J. MacLullich, S. D. Shenkin, and T. J. Quinn, "Predicting Discharge to Institutional Long-Term Care After Stroke: A Systematic Review and Metaanalysis," *J Am Geriatr Soc*, vol. 66, pp. 161–169, Jan 2018.

[3] D. S. Ouellette, C. Timple, S. E. Kaplan, S. S. Rosenberg, and E. R. Rosario, "Predicting discharge destination with admission outcome scores in stroke patients," *NeuroRehabilitation*, vol. 37, no. 2, pp. 173–179, 2015.

[4] J. K. Harrison, K. E. Walesby, L. Hamilton, C. Armstrong, J. M. Starr, E. L. Reynish, A. M. J. MacLullich, T. J. Quinn, and S. D. Shenkin, "Predicting discharge to institutional long-term care following acute hospitalisation: a systematic review and meta-analysis," *Age Ageing*, vol. 46, pp. 547–558, Jul 2017.

[5] M. W. Temple, C. U. Lehmann, and D. Fabbri, "Predicting Discharge Dates From the NICU Using Progress Note Data," *Pediatrics*, vol. 136, pp. 395–405, Aug 2015.

[6] Y. Yang, P. Xie, X. Gao, C. Cheng, C. Li, H. Zhang, and E. Xing, "Predicting Discharge Medications at Admission Time Based on Deep Learning," *ArXiv e-prints*, Nov. 2017.

[7] A. Avati, K. Jung, S. Harman, L. Downing, A. Ng, and N. H. Shah, "Improving Palliative Care with Deep Learning," *ArXiv e-prints*, Nov. 2017.

[8] E. Choi, M. T. Bahadori, A. Schuetz, W. F. Stewart, and J. Sun, "Doctor ai: Predicting clinical events via recurrent neural networks," in *Proceedings of the 1st Machine Learning for Healthcare Conference* (F. Doshi-Velez, J. Fackler, D. Kale, B. Wallace, and J. Wiens, eds.), vol. 56 of *Proceedings of Machine Learning Research*, (Children's Hospital LA, Los Angeles, CA, USA), pp. 301–318, PMLR, 18–19 Aug 2016.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[10] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

[11] J. Mount and N. Zumel, "vtreat: A statistically sound 'data.frame' processor/conditioner." https://cran.r-project.org/web/packages/vtreat/index.html, 2018.

[12] A. Liaw, M. Wiener, *et al.*, "Classification and regression by randomforest," *R news*, vol. 2, no. 3, pp. 18–22, 2002.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[14] S. Nair, O. Moindrot, and G. Genthial, "cs230-code-examples." https://github.com/cs230-stanford/cs230-code-examples/tree/master/pytorch/vision, 2017.

[15] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[16] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *ICML*, 2015.

[17] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[18] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[19] A. E. Johnson, T. J. Pollard, L. Shen, L.-w. H. Lehman, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. Anthony Celi, and R. G. Mark, "Mimic-iii, a freely accessible critical care database," *Scientific Data*, 2016.