Predicting Mortality of ICU Patients using Neural Networks

John Evans, Rajesh Pothamsetty, Mengnan Zhang
Georgia Institute of Technology - BDH6250, Atlanta, Georgia, United States

Abstract

The Intensive Care Unit (ICU) provides support to the most severely ill patients in the hospital, offering vital, life saving treatments. Predicting mortality in patients hospitalized in the ICU is crucial for assessing severity of illness and adjusting the treatments and intervention plans.

The Multi-parameter Intelligent Monitoring in Intensive Care (MIMIC) database has been built up and maintained by the Laboratory of Computational Physiology at the Massachusetts Institute of Technology. By exploring existing features such as age, sex, SAPS II, and SOFA scores, the goal of this project is to correlate these features with patient mortality outcomes using a combination of feature engineering, Deep Learning, and hyperparameter tuning strategies to better predict ICU mortality.

A video presentation of this paper is publicly available at: https://youtu.be/QvwpUlsPMcc

1 Introduction

Electronic monitoring systems and health records facilities generate large amounts of information every day. Converting this information into knowledge improves the efficiency of clinical decision making as well as the overall quality of care.The Intensive Care Unit (ICU) provides support to the most severely ill patients in the hospital, offering vital lifesaving treatments. Patients admitted to the ICU suffer from severe illnesses and/or trauma, and are at extreme risk of dying. Predicting mortality in patients hospitalized in the ICU is crucial for assessing severity of illness and adjusting treatments and intervention plans. ICU mortality rates differ from the underlying disease process, with death rates as high as 25%[1]. The ICU is not only a place with a high-death rate, but it is also an environment with ever-rising costs and heavily constrained resources. 25% of U.S. health care spending goes to 6% of the people who die every year. The ICU accounts for 20% of all health care costs[2].

Effective predictors of Mortality have the potential to identify high-risk patients and improve ICU resource allocation. By predicting the time frames with high mortality risk, hospital ICUs can have a better idea of resource allocation during a patients' stay. More importantly, patients can receive better treatment options and more efficient care during their stay.

Mortality risk can be approximated by evaluating the severity of a patient's illness. Illness severity is determined by evaluating a combination of physiologic, clinical, and demographic determinants. Several severity of illness (SOI) scores have been introduced in the ICU to predict mortality. Examples of SOIs include the Acute Physiology and Chronic Health Evaluation (APACHE), the Simplified Acute Physiology Score (SAPS), and the Sequential Organ Failure Assessment (SOFA). This project explores SAPS, with the potential to add SOFA to future iterations as a part of its predictive model.

SAPS II score is used to score the ICU patients based on 17 variables, including 12 physiological variables, age, type of admission (scheduled surgical, unscheduled surgical, or medical), and three underlying disease variables (acquired immunodeficiency syndrome, metastatic cancer, and hematologic malignancy)[4]. The SAPS II score ranges from 0 to 163, with 116 points assigned to physiological variables, 17 points for age, and 30 points from previous diagnosis. Le Gall et al[5] developed and validated SAPS II to provide a method to convert the score to a probability of hospital mortality without having to specify a primary diagnosis in the 13152 patient samples. The probability of death is calculated using logistic regression based on the SAPS II score.

SOFA score was developed to quantify the severity of patient illness based on the degree of organ dysfunction data on

six organ failures. Each type of dysfunction is scored on a scale of 0-4[6]. One failure plus a respiratory failure indicates the lowest mortality. All the other combinations yield mortality between 65% and 74%.

In this paper, we introduce a predictive modeling approach for predicting ICU mortality utilizing a Deep Learning Neural Network that consumes both raw and engineered features from the Multiparameter Intelligent Monitoring in Intensive Care (MIMIC) III database provided by the Laboratory of Computational Physiology at MIT[9]. We begin in Section 2 by surveying related work in the medical field. From there, we discuss our method in Section 3, and provide our final results in Section 4. Finally, we discuss some lessons learned and possible future improvements in Section 5, and conclude in Section 6.

2 Related Work

There is considerable research that focuses on metrics such as length of stay (LOS), and severity of illness to predict ICU mortality. While some of this work has been modeled as Logistic Regression, most of the research has relied on binary classification as a means of mortality prediction. Since the commoditization of GPU computing, more researchers are adopting deep learning approaches where Logistic Regression and Binary Classification are used as the baseline to beat.

Phenotyping is a much broader approach compared to Logistic Regression or Binary Classification, and it has been demonstrated by Lasko et al[8], that feed-forward neural networks can provide better results than these approaches. There isn't an agreed upon "gold standard" of benchmarks for evaluating approaches. Harutyunyan et al[10] have proposed several benchmarks for evaluating competing models including in-hospital mortality, and acute care phenotype classification. We use the results from these benchmarks as front-loaded features applied to our predictive model in addition to the features we have extracted from MIMICIII.

A final approach related to this project involves the construction of Recurrent Neural Networks for the purpose of prediction. While wide gains have been made in medical research involving Deep Learning, there seems to be little research that involves RNNs for the purpose of predicting ICU mortality. That being said, Purushotham et al[11] have employed an adversarial RNN to perform domain adaptation between different patient age groups for both mortality prediction, and diagnosis classification.

3 Method

This section outlines a prototype of our solution at the time of this writing. It in no way resembles a complete solution, and is undergoing rapid iteration. That being said, the overarching bigdata pipeline can be visualized in three distinct steps; feature selection, training, and prediction.

3.1 Feature Selection

The dataset for our models contains 42,276 patients with 1:10 imbalanced classes. For the purposes of this project we selected seven distinct categories of features which we outline in the following table:

From the table we see that medication, and diagnoses are raw counts, while chart events, LOS, and age are raw values. Additionally, we compute the SAPS, and SOFA scores by running a script provided in the mimic-code repository by MIT[12].

To compute the Phenotype score, we first run the feature extraction code provided by Harutyunyan et al[10,14] to generate one diagnoses.csv file per patient. The Phenotype score is the count of the twenty-five most commonly identified conditions in the ICU. These conditions are further broken down by severity into twelve critical, eight chronic, and five recurring or chronic conditions. We identified five distinct last care units associated with a specific ICUSTAY_ID that could contribute to the overall mortality of the patient. We transformed these into binary features which we then sum up.

Table 1: Feature Categories per ICUSTAY_ID

Category	Count of Features
Chart Events with Numerical Score	6463
Medication/ICD9Cumulative ICUStay/Abnormal Lab Values Count	3
Binary Last Care Unit	4
Length of Stay (LOS)	1
Acute Care Phenotype Score	1
SAPS	17
Ecomorbidities	30
SOFA	7

Next, we compute the ecmorbidites using a script provided in the mimic-code repository by MIT[12] and count up the lab events for each patient. Since these features aren't directly associated with an ICUSTAY_ID, we instead use the HADM_ID to map onto the ICUSTAY_ID.

One caveat to this approach is that chart events contain a high number of missing values due to the fact that different patients have different chart events with different diagnoses for each ICUSTAY_ID. We address this by including a binary feature to help account for missing data. A 1 represents a missing value, while a 0 represents a non-missing value. Finally, we mask the missing values with a random negative integer so that they aren't included in the final computation. This ensures that missing data is omitted, while non-missing values are included without a loss in generality[13].

Finally, we map the Mortality_In_Hospital_all_stays.csv from MIMICIII as our classification label. The EXPIRE_FLAG indicates whether a patient has died, and includes both patients that have died in and outside of the ICU. We only focus on patients with an EXPIRE_FLAG associated with a death in the ICU.

Once these values are mapped, we extract each of these features using Spark, and then store the resulting featureset to a text file. The featureset is then passed to our neural network for the training and prediction steps.

3.2 Training and Prediction

Our neural network architecture is an ensemble of various techniques built on top of Keras that feed into a Logistic Regression. The ensemble itself is comprised of the following components:

1. A DNN that learns chart events.
2. A NN that learns regular features from the featureset.
3. Extracted predictions from a Long Short Term Memory Recurrent Neural Network.

In regards to three, we are once again building off of the work done by Harutyunyan et al[10] by utilizing their resulting mortality probability predictions. Each prediction is derived from a timeseries of stays in the ICU spanning forty-eight hours. Unfortunately, the LSTM is designed to train and validate on a limited subset of the dataset. To architect against this, we first pre-trained the model, then validated on a combination of the training and validation sets. Normally this would overgeneralize the model; however, in this context we are mostly interested in the prediction's probability for a given ICU stay, opposed to the prediction's label.

The overarching model can be thought of as a Context Based Weighting system, as illustrated in the figure below:

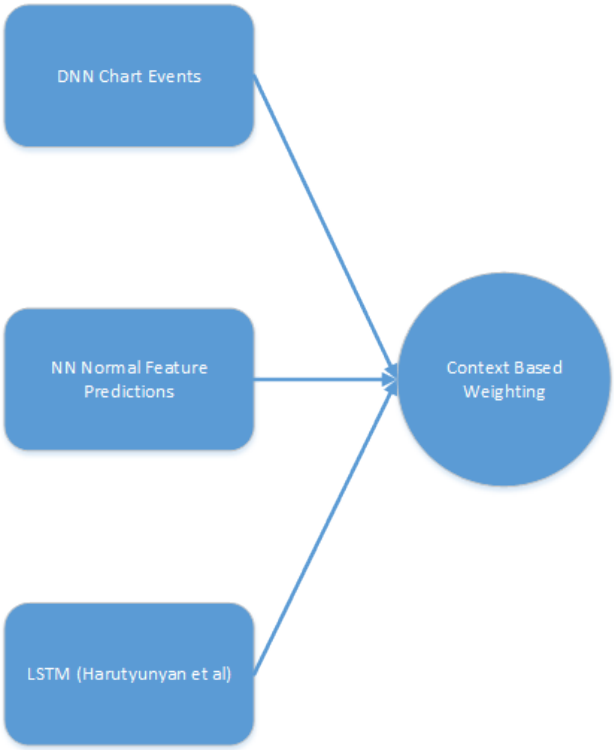

Figure 1: Context based Architecture

The DNN is comprised of five hidden layers where the number of units decreases from 3000 down to 50 before reaching the output layer.This design was determined empirically by experimenting with both wider, and deeper layers. We concluded that more layers did nothing to enhance the performance of the model, and fewer units were unable to provide an effective generalization.

Each hidden layer uses a ReLU activation function, and the output layer uses a Sigmoid function for classification. Note; however, that there are many dense connections between layers early on that will lead to significant overfitting. To combat this issue, a regularization technique known as dropout has been introduced both in the input layer, as well as the first two hidden layers. Dropout deactivates 20% of the inputs on the layer where it is applied, transforming the dense connections in Figure 1, to something like the illustration below:

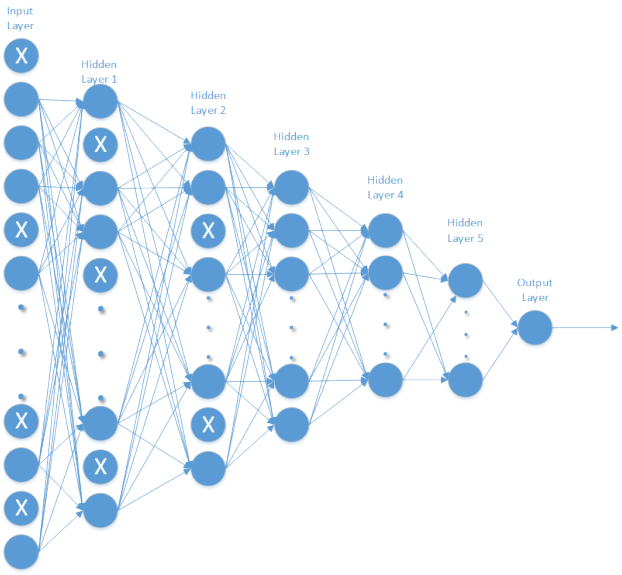

Figure 2: DNN with Dropout Applied

This allows for much better generalization, and helps to prevent overfitting. The output layer emits either a zero or a one. A one indicates that the model has predicted the patient is dead, while a zero indicates a prediction of alive. We validate the model utilizing five fold stratified cross-validation.

3.3 Hyperparameter Tuning

We explore two well know optimizers; Stochastic Gradient Descent, and Adam. Adam is an Adaptive Learning Rate Method that can be used instead of SGD to update network weights based on training data. To initialize our weights we have chosen to use the glorot-uniform method. Additionally, we have selected weighted binary-crossentropy with 1:10 weights as our loss function. This is because alive ICU events appear ten times more frequently than ICU events that result in death. One of the challenges of architecting an effective neural network is the fact that there are numerous hyperparameters to tune such as momentum, decay, and number of epochs. Rather than applying simple guess and check to tune these parameters, we have implemented a brute force approach known as Grid Search to evaluate numerous model configurations, and select the best model. After testing a few different scenarios, we decided to use Adam due to its high adaptability. For the LSTM model, we used the same set of hyperparameters used by Harutyunyan et al to achieve their optimal results[10]

4 Experimental Results

Our initial model has the following configuration:

Table 2: Baseline Configuration

Parameter	Value
Optimizer	Adam
Epochs	10
Batch Size	20
Learning Rate	0.001
Decay	0.9
Momentum	0.999
CV Folds	5

The total execution time of the model was approximately 18 hours, and produced the following accuracies for each piece of the model: As we see from the charts, the DNN is particularly effective at learning chart events and signifi-

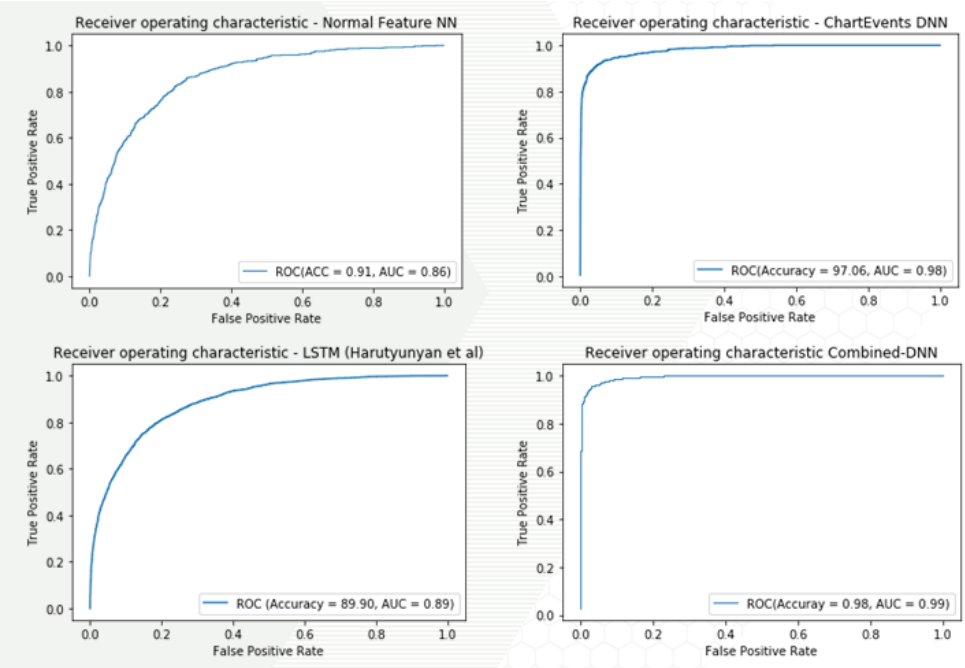

Figure 3: Model Results

cantly outperforms both the LSTM benchmark, and the NN. Intuitively, this means that the DNN appears to be better at predicting both true positive (dead patients) and true negative (alive patients). What's interesting is that the reason why both the NN and DNN outperform the LSTM is due in large part to the fact the LSTM appears to have trained to be a strong predictor of alive patients, but is particularly ineffective at predicting dead patients. This is due in large part to the fact the LSTM trains on 17 physiologic time series variables that only represent a subset of what is included in Physionet/CinChallenge 2012 dataset[?]. Conversely, we have trained the DNN and NN components of the model on more than 35,000 ICU stays for which all of the events were recorded.

The best approach to using these models in predicting mortality is to identify the context in which their performance is optimal. For the first twenty-four to forty-eight hours in the ICU, the DNN doesn't have sufficient data to provide a good prediction. On the other hand, the LSTM looks specifically at the order of events while in the ICU, and as such is capable of providing a much stronger prediction when very little data is available. This means that the longer a patient stays in the ICU, the more weight that should be placed on the prediction of the DNN and vice-versa. Further research should be done to identify the context or timeframe of an ICU stay for using these models coherently.

5 Lessons Learned and Future Improvements

Prototyping a neural network isn't without its challenges. Like any Computer Science problem, the model consumes both time and space. The more layers you have, and the more units in each layer, the longer it takes for neural network training and evaluation and the more memory that is consumed. One major challenge that this project faced was that the neural network would oftentimes consume too much memory and trigger Out of Memory Errors on the GPU.

This issue, while challenging, isn't without its remedies. The main reason why these out of memory issues would occur is because the product of the high number of features, and the number of units in a hidden layer was very large.

For example, from the input layer with 6515 features to the first hidden layer with 3000 units is 6515x3000 matrix, or approximately 20M elements. Many of these features are sparse by design, and so our dense representation of the the featureset is a very inefficient use of space. Converting our dense representation into a sparse representation should greatly improve memory efficiency and significantly reduce the likelihood of failures due to out of memory errors.

Another improvement that our approach could benefit from is the concept of batching the inputs being fed into the network. Keras allows developers to specify a batch size which we have done, but no attempts have been made to break up the the featureset into batches. This means that an inordinate amount of RAM is required at load time even though the featureset itself is a little over 1gb.

A third possibility for improvement is in the construction of the timeseries. Currently, the LSTM timeseries data leverages only seventeen variables for predictions. Adding more variables may improve the overall predictability of the LSTM model. Additionally, the inclusion of more features that are similar to the Cumulative ICU Stay Count could improve the overall performance of the Normal Features NN. Applying these two improvements would help to bring balance to the model, and help to provide an ideal context based weighting.

Finally, the implementation has no checkpointing. This means that if a task fails for any reason, the entire model is lost, and must be retrained. Moreover, there is no way to experiment with different numbers of epochs without completely retraining the model. Keras provides a callback system that can be taken advantage of to checkpoint the model at each epoch. Should a task fail, a user can simply read in the model at the last successful epoch.

6 Conclusion

In this paper we outlined an approach to ICU mortality prediction using an ensemble of Deep Learning, LSTM RNN, and traditional NN techniques built with Keras running on top of Tensor Flow. Our model achieved an overall accuracy of 97.6% with an AUC of 98.0% on ten epochs, improving upon the work done by Harutyunyan[11] due in large part to the fact the LSTM model only trains on a very small number of dead patients, and so tends to bias towards living patients. Finally, we discussed some potential for improvements including better memory management through batching, and implementing callbacks for checkpointing.

References

1. Dasgupta, S., et al., Nosocomial infections in the intensive care unit: Incidence, risk factors, outcome and associated pathogens in a public tertiary teaching hospital of Eastern India. Indian Journal of Critical Care Medicine : Peer-reviewed, Official Publication of Indian Society of Critical Care Medicine, 2015. 19(1): p. 14-20.
2. Aldridge, M.D. and A.S. Kelley, The Myth Regarding the High Cost of End-of-Life Care. American Journal of Public Health, 2015. 105(12): p. 2411-2415.
3. Knaus, W.A., et al., APACHE II: a severity of disease classification system. Crit Care Med, 1985. 13(10): p. 818-29.
4. Le Gall, J.R., et al., Mortality prediction using SAPS II: an update for French intensive care units. Critical Care, 2005. 9(6): p. R645.
5. Le Gall, J., S. Lemeshow, and F. Saulnier, A new simplified acute physiology score (saps ii) based on a european/north american multicenter study. JAMA, 1993. 270(24): p. 2957-2963.
6. Rapsang, A.G. and D.C. Shyam, Scoring systems in the intensive care unit: A compendium. Indian Journal of Critical Care Medicine : Peer-reviewed, Official Publication of Indian Society of Critical Care Medicine, 2014. 18(4): p. 220-228.
7. Marafino, B.J., W. John Boscardin, and R. Adams Dudley, Efficient and sparse feature selection for biomedical text classification via the elastic net: Application to ICU risk stratification from nursing notes. Journal of Biomedical Informatics, 2015. 54: p. 114-120.
8. Lasko, T., et al., Computational Phenotype Discovery Using Unsupervised Feature Learning over Noisy, Sparse, and Irregular Clinical Data. PLoS ONE 8. 2013: p. 2-3.
9. Saeed, M., et al., Multiparameter Intelligent Monitoring in Intensive Care II (MIMIC-II): A public-access intensive care unit database. Critical care medicine, 2011. 39(5): p. 952-960.
10. Harutyunyan, H., et al., Multitask Learning and Benchmarking with Clinical Time Series Data: In Proceedings of ACM Conference. 2017: p. 2-4.
11. Purushotham S, et al., Variational Recurrent Adversarial Deep Domain Adaptation. International Conference on Learning Representations. 2017: 2-5.
12. https://github.com/MIT-LCP/mimic-code/blob/bec1470b33352117ba00c62768c94c5ac93d9996/concepts/. April 2018
13. https://docs.scipy.org/doc/numpy-1.13.0/reference/maskedarray.generic.html#rationale. April 2018.
14. https://github.com/YerevaNN/mimic3-benchmarks/blob/master/scripts/extract_subjects.py. April 2018.