Deep Learning Risk Prediction of Bloodstream Infection in the Intensive Care Unit

Zachery Boner
zwb6kg@virginia.edu
University of Virginia
Charlottesville, Virginia, USA

Christopher C. Moore
ccm5u@virginia.edu
Division of Infectious Diseases and International Health
Department of Medicine
University of Virginia
Charlottesville, Virginia, USA

N. Rich Nguyen
nn4pj@virginia.edu
Department of Computer Science
University of Virginia
Charlottesville, Virginia, USA

ABSTRACT

Bloodstream infection is a leading cause of mortality in patients in the intensive care unit (ICU). Early detection and treatment of bloodstream infection is associated with significantly better clinical outcomes. However, the detection of bloodstream infection requires a time-consuming blood culture. Far more blood cultures are ordered than return positive; this excess of blood cultures can cause delays in result and therefore treatment. In this study, we aim to move towards a continuous monitoring tool to help doctors and nurses in the ICU identify which patients require a blood culture, with the goal of supporting earlier detection and treatment of bloodstream infection. We formulated this goal as a multivariate time-series classification problem and applied powerful predictive deep learning approaches to model multivariate time-series data. We used a variety of model validation and explainability techniques to help understand the decisions of these deep learning models and promote trust in their predictions.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Supervised learning by classification**; *Cross-validation*; Feature selection; • **Applied computing** → **Health informatics**.

KEYWORDS

deep learning, data mining, datasets, explainable AI

ACM Reference Format:
Zachery Boner, Christopher C. Moore, and N. Rich Nguyen. 2022. Deep Learning Risk Prediction of Bloodstream Infection in the Intensive Care Unit. In *KDD Undergraduate Consortium, August 14-18, 2022, Washington, D.C.*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Bloodstream infections are associated with high risk of mortality, long hospital stays, and expensive treatment [17]. Patients in the intensive care unit (ICU) are at especially high risk of bloodstream infection given their already critical conditions and the common usage of intravenous catheters in their treatment. The most common pathogens related to bloodstream infection are bacteria and other microbes. The primary treatment option is broad-usage antibiotics, which are becoming decreasingly effective as general antibiotic resistance grows. Furthermore, the detection and diagnosis mechanism for bloodstream infection is a blood culture [1], which has a turnaround time of up to several days and a risk of contamination, invalidating the result [2] [19].

Bloodstream infection remains a difficult disease to identify clinically. Previous studies have demonstrated good results predicting the presence of bloodstream infection [15], predicting the outcomes of patients with bloodstream infection [22], and applying predictive modeling approaches to identifying pathological signatures of bloodstream infection [21]. In this study, we developed a predictive model for bloodstream infection and then extended that result towards a continuous monitoring tool for evaluating risk of bloodstream in a time-aware environment. This continuous monitoring is designed to inform doctors and nurses in the ICU when to draw blood cultures and when to begin antibiotic treatment.

Our candidate predictive models for bloodstream infection prediction were deep neural networks. We considered three main classes: feed forward networks, recurrent neural networks (RNNs), and convolutional neural networks (CNNs). We found that RNNs and CNNs outperformed our baseline feed forward networks and achieved significant predictive power in their results. We extended our modeling results towards clinical usefulness in a continuous monitoring setting by "sweeping" our model's predictions over a longer time window. In this way, we simulated a real clinical environment in which the model outputs a changing fold risk score.

2 DATA

2.1 Data Pipeline

In order to establish a consistent data acquisition, preprocessing, modeling, and explainability process, we built a modular pipeline for each of these stages, with each stage producing output that fit directly into the next stage. In such a way, we maintained an understanding of our data at all times, could easily make changes to any stage of the process in a controlled way, and could quickly produce and reproduce experiments. In our data acquisition stage, we combined raw comma-separated-value (CSV) files containing data directly from University of Virginia Electronic Medical Records (EMRs) about individual patients into a single data pool and reformatted the data as a TensorFlow dataset to optimize training with

KDD Undergraduate Consortium, August 2022, Washington, DC, USA

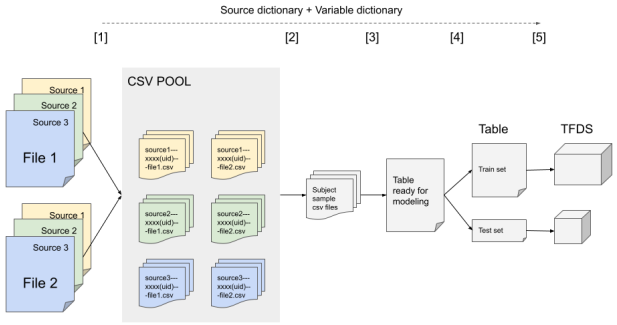


Figure 1: Broad pipeline flow of data from patients at risk of bloodstream infection in the intensive care unit at the University of Virginia, 2011-2015

TensorFlow implementations of our deep learning architectures. The overall flow of our data processing pipeline is demonstrated in Figure 1.

2.2 Data Description

The data used in our experiments was sourced from the University of Virginia Electronic Medical Record. The population used to train our machine learning models was a collection of 3969 patients and 4582 unique blood culture episodes from 2011-2015. With 33 features and 48 timesteps in each episode, this corresponds to 220 thousand rows of data or 7.25 million hourly measurements.

We defined an episode to consist of hourly sampled laboratory and vital sign data in the 48 hour period leading up to the acquisition of a blood culture. As control episodes, we included random 48 hour periods with no blood culture acquisition in the dataset and labelled them as negative results. Therefore, our response variable had two classes: a positive class containing patients with positive blood cultures, and a negative class containing both cases where a negative blood culture result was received and cases where no blood culture was obtained.

These episodes consisted of 33 common laboratory and vital signs (description in appendix B) tracked on nearly all patients that are admitted to the ICU. From this cohort, 13.1% of the episodes were associated with a positive blood culture result and 66.3% with a negative blood culture result. The remaining 20.5% of the episodes did not have a blood culture drawn at all. This data was included so that the risk probability output of the model was not conditional on a blood culture being drawn. Full counting statistics for each of our datasets is presented in Table 1.

2.3 Data Preprocessing

For consistency and reproducibility, we developed a thorough preprocessing pipeline for our data; this pipeline takes as input tabular data for individual patients and outputs a dataset in the TensorFlow Dataset format of "episodes" of bloodstream infection. Our data is stored in raw form as individual CSV files for each patient. We processed these CSV files, split them into episodes, then combined the episodes into a single multidimensional dataset. During this stage, we removed any episodes where the blood culture results showed common contaminating organisms, as well as repeat blood cultures for a patient within 7 days following a positive culture. We maintained at all times a file with alternative names, unit information, cutoff values for outliers, and human-readable text regarding how to interpret the data.

Table 1: Number of episodes and proportion of dataset for our entire dataset, as well as training/validation/testing; outcomes of negative, no blood culture drawn, and positive.

		total	negative	none	positive
total	count	6557	4319	1361	877
	ratio		65.8%	20.7%	13.3%
train	count	4582	3039	940	603
	ratio		66.3%	20.5%	13.1%
validation	count	1320	852	283	185
	ratio		64.5%	21.4%	14.0%
test	count	655	428	138	89
	ratio		65.3%	21.0%	13.5%

We used outlier cutoff values from a previous study within our group, performed by Zimmet, et al. [21]. We then performed forward imputation on each patient a maximum of 24 hours into the future. We filled in any remaining missing values via global feature-wise median imputation. Our approach to filling in missing values was based on two core assumptions: first, that variables that are infrequently sampled (i.e. less than one sample per hour) were unlikely to change much between samples, so forward imputation made sense; second, that after a 24 hour time period, forward imputation was no longer applicable. For any time period of missing data greater than 24 hours, we defaulted to the global median value for that variable to avoid a biased assumption.

After the preparation stages, we shuffled the data into training, validation, and test sets at a 70/20/10 ratio. We chose a batch size of 64 episodes for our model training, validation and testing.

3 METHODS

3.1 Model Training

We performed our modeling experiments using the Keras Sequential framework in TensorFlow. We used Keras components and layers to construct and optimize model architectures and hyperparameters. Our experiments were run on the UVA Computer Science Department's computing cluster. The specifications of the machines used for training were as follows: Forty Intel Xeon 4210 CPU cores (2.2GHz), Four Quadro RTX 4000(8GB) GPUs, and 512GB of main memory.

We explored three primary classes of models in our experimentation: feed forward neural networks, recurrent neural networks, and convolutional neural networks. Our modeling experimental approach was as follows:

(1) Begin with a small instance of each model.
(2) Scale the size of the model up incrementally until performance on the held out validation set stops increasing.

(3) Perform hyperparameter optimization of the large model.
(4) Repeat steps 2 and 3 until no performance gains are realized.

This approach had a few important advantages. We only performed expensive hyperparameter optimization when necessary. Further, by scaling up the model size incrementally, we biased for simpler models and helped to mitigate overfitting.

3.2 Deep Learning Approaches

We now describe the structure and distinguishing features of each class of model that we experimented with, and discuss briefly why we chose each class.

3.2.1 Feed Forward Neural Networks. Feed Forward neural networks are a classical baseline modeling approach for many types of data. They are dense, in that they consider the output as a linear combination of every data point for all variables. Importantly, feed forward networks do not consider the data as a time series, rather merely as a collection of data. They do not assign order or maintain a notion of time dependence in their architecture. We chose to use this class as described – as a baseline to evaluate whether or not our deep learning architectures, which do consider order and time dependence in their architectures, are able to outperform feed forward networks.

3.2.2 Recurrent Neural Networks. Recurrent neural networks establish time dependence in their architectures by creating a dense feed forward layer for every time step, then calculating gradients during training across those layers. This is coupled with an additional hidden layer. This architecture helps to mitigate gradient explosion and vanishing, and can learn relationships on long time scales. Other experiments have demonstrated that the GRU achieves similar performance to the Long-Short Term Memory (LSTM) architecture, with a smaller training cost [7]. For this reason, we decided to focus our experimentation on the GRU instead of the LSTM.

3.2.3 Convolutional Neural Networks. Convolutional Neural Networks (CNNs) were originally designed for image processing, but have found substantial application in the multivariate timeseries learning domain [10]. CNNs work by using a weighted filter passed over blocks of pixels, or in our case time steps and features, and then performing a pooling operation to combine the output of the filters into compacted encodings of the input data. By using multiple convolutional layers, patterns can be learned over the entire feature space and over long time periods.

3.3 Regularization and Training Optimization

During model training, we employed several methods to promote stability and generalization of the models. We used the *nadam* optimizer, which is the well-known *adam* optimizer with Nesterov momentum [9]. We included dropout layers before high-parameter count layers in our models as regularization. We utilized validation during model training to identify overfitting in real time. By integrating the framework Weights and Biases into our training and experimenting loop [3], we tracked several metrics, which were evaluated on the training set and the validation set after every epoch: these include binary crossentropy loss, AUC, precision, recall, and raw accuracy, as well as true positive, false positive, false negative, and true negative counts.

The Area Under the Receiver Operating Characteristic (AUC) is a measure of the tradeoff between true positive rate and false positive rate of the model at different evaluation thresholds. An AUC between 0.70 and 0.80 is considered acceptable, an AUC between 0.80 and 0.90 is indicative of a significantly predictive model, and an AUC above 0.90 is excellent. During training, if the validation AUC did not increase for 20 consecutive epochs, we halved the learning rate. If validation AUC did not increase for 30 consecutive epochs, we ended training to save computational resources and time.

Given the size of our data and the use of deep learning models, we found that performing traditional hyperparameter optimization approaches was too expensive in computational time. Therefore, we focused primarily on iterative, human-driven hyperparameter optimization. For each learning architecture we iteratively scaled up the parameter count of each model until performance stopped increasing. At this point, we began optimizing non-structural hyperparameters, such as learning rate, factor of learning rate decrease, and early stopping thresholds. We found that in general so long as the initial learning rate was low enough for stable training, the other hyperparameters made little to no difference in performance. We chose an initial learning rate of 0.0001, a factor of learning rate decrease of 0.2, and an early stopping threshold of 30 iterations without improvement. Details of structural hyperparameters, such as layer count and width, are provided for our best model from each class in appendix A.

4 RESULTS

To evaluate and explain our models, we measured performance on the previously stated metrics on a test set held out from both training and validation. We also evaluated fold risk on time windows approaching time of blood culture on models trained on only 24 hour episodes of data. We measured model predictions of probability of positive outcome, divided by the proportion of positive outcomes in the data set, on the time windows [-47, -24] through to [-23, 0] for each episode then plotted these fold risk scores as a time series. We randomly sampled these predictions for visual inspection, to check that the time series matched clinical expectations on known positive and negative cases. To provide understanding of the clinically relevant features that align with current evidence based medical practice, we calculated feature importance in our predictive models [21].

Table 2: Performance statistics of various deep learning models, evaluated on the test set

	Precision	Recall	AUC
FFN	0.443	0.448	0.767
CNN	0.505	0.516	0.828
GRU	0.558	0.483	0.835

4.1 Modeling Results

In our modeling experiments, we considered three main classes of models: feed forward neural networks, convolutional neural networks, recurrent neural networks [5]. We evaluated model performance using a variety of metrics: binary cross-entropy loss,

AUC, precision, recall, and raw accuracy, as well as true positive, false positive, false negative, and true negative counts. Our primary evaluation metric among these was the AUC. We evaluated each of these metrics on the validation set after every epoch of model training. This allowed us to identify overfitting during long training runs and to automatically halt training early when it became clear that validation AUC was no longer improving.

The model we trained with the highest validation AUC score was the GRU model, with 0.835 AUC. CNNs achieved similar performance with 0.828 AUC. Feed forward networks performed adequately, although they had a significantly lower AUC score than both GRUs and CNNs. Both GRUs and CNNs outperformed baseline the feedforward networks by a significant margin. See Table 2 for a performance breakdown of each approach.

At first glance, our precision (positive predictive value) and recall (sensitivity) scores appear low. However, in the clinical setting, these scores are powerful. As seen in Table 1, 13% of blood cultures return positive. This indicates a precision of 0.13 on behalf of clinicians. Our GRU model attains about 0.56 precision. This corresponds to a large reduction in the rate of false positive blood cultures [1]. The clinicians' recall, however, is substantially higher than our models. Our models attain about 0.50 recall; the clinician's goal is to approach a recall of 1.0, so that there are no false negatives. However, we can increase recall at the cost of precision by changing our positive risk prediction threshold; at 0.90 recall, for example, the GRU model still attains 0.20 precision. This suggests that our model can perform competitively in prediction of bloodstream infection.

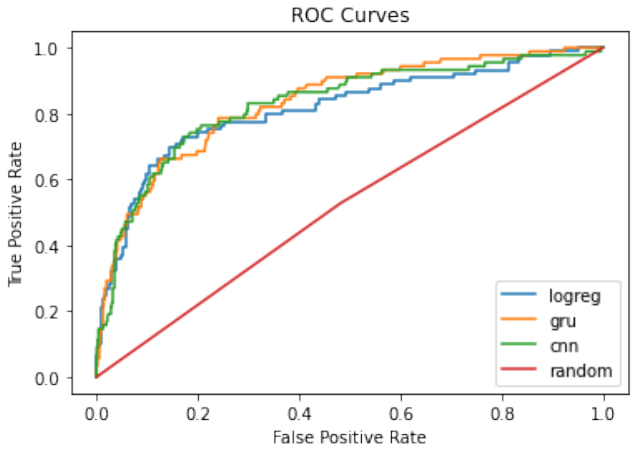


Figure 2: ROC curve of the best models in each class, compared to a random guesser

4.2 Continuous Monitoring

After we developed sufficiently predictive models on 48 hour time windows, we took another step towards evaluating our approach's clinical viability [11]. We made a new dataset consisting of only the last 24 hours of each 48 hour time window, then trained the best architectures from our earlier work on this new dataset. With these trained models, we returned to the 48 hour time window dataset and swept over 24 hour subsets of this data to obtain a model output for 24 time steps in sequence, leading up to the time of blood culture. This enabled us to evaluate risk dynamically over time.

We divided our model risk probability predictions by the baseline probability that a patient in our dataset would have a positive blood culture, equal to the proportion of positive blood cultures in our dataset. This gave us a fold risk score. A fold risk score at a time step t was considered high risk if it was above 3 [14]. A fold risk score of 3 or above means that the patient is 3 or more times more likely than an average patient to return a positive blood culture at that time step.

In Figure 3, we present the mean fold risk predictions at each time step for ground truth positive patients (blue) and ground truth negative patients (red). The band surrounding the mean predictions indicates the standard error of the mean for each class of patients. Our model, on average, effectively discriminates between positive and negative patients. Also notice that, on average, the model's fold risk predictions for positive patients increased as the time window neared the time of blood culture (time zero). In contrast, the fold risk predictions for negative patients remained low and stable.

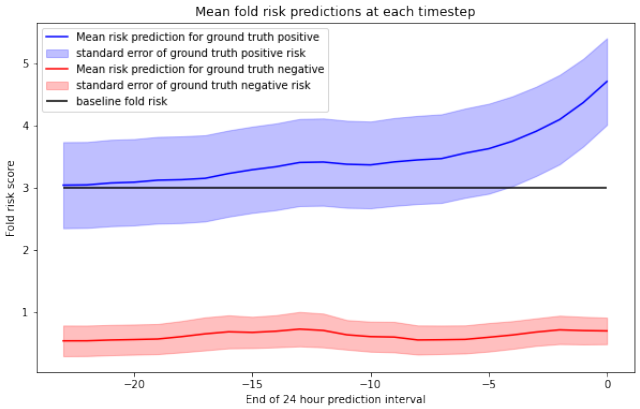


Figure 3: Mean fold risk predictions for 24 hour windows.

In the Figure 4a, we examine specific examples, demonstrating four different fold risk scores for individual patients, two positive and two negative. Notice that the risk scores for these patients increases over time for the positive patients, and also that this behavior occurs well before time zero. This indicates that, for these patients, the increase in fold risk over time could have resulted in a blood culture being drawn earlier and a potentially better outcome for the patient.

It is important to note, however, that our model does not always display such ideal discriminating ability. There are both false positive and false negative examples, as demonstrated in Figure 4b. We similarly display four different fold risk plots, again two positive and two negative, indicating the sorts of failures we might observe. Some negative patients are predicted as high risk, and some positive patients are predicted as low risk.

Our results suggest that our approaches with deep learning could yield predictive, potentially clinically useful results. Our AUC scores of 0.8 and higher indicate predictive power, and the fold risk scores plotted over time behave in a clinically useful way. Patients who

returned ground truth positive blood cultures show fold risk scores above 3 and increasing over time. An increasing fold risk score over time indicates to a doctor or nurse that a patient's condition is worsening towards bloodstream infection. This information can inform a clinician's decision to draw a blood culture. On the other hand, a steady low risk score indicates to clinicians that the patient is unlikely to have a bloodstream infection, and therefore does not need a blood culture drawn. Thus, our approach could help improve the cost and efficiency of care while improving outcomes.

4.3 Feature Importance

To examine feature importance, we used Shapely Additive Explanations (SHAP values) to evaluate feature importance in our models [16]. The SHAP value is a natural way to represent the contribution of individual features to the predictions of a model. We consulted clinicians in our group to determine if the features most important to a model made sense from a physiological standpoint. In addition to the known important features to bloodstream infection (Temperature, Blood Pressure, and Heart Rate) we identified Blood Urea Nitrogen (bun), Albumin, Platelet count, Chloride, Creatinine, and Phosphorus as important features. A plot of these Shapley values for our best-performing GRU model is presented in Figure 5. Blue and red points indicate below-mean and above-mean feature values, respectively. Orientation on the x-axis indicates impact on model output, or importance.

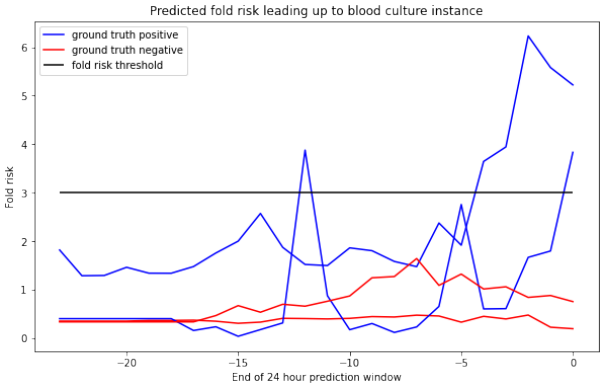
5 DISCUSSION AND RELATED WORK

There are three main problem domains that we are working within: the prediction of bloodstream infection, the general study of multivariate timeseries classification, and the use of deep learning for multivariate timeseries classification. We discuss each notion, along with related work, in the following text.

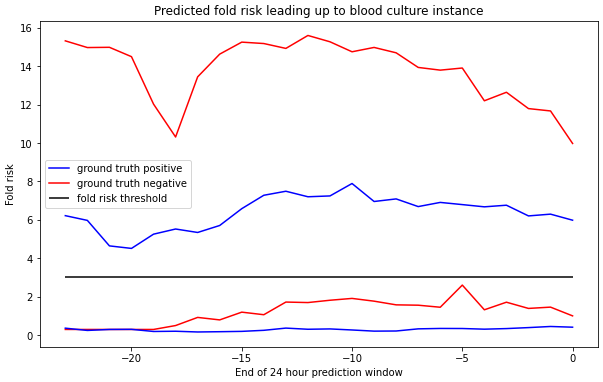
5.1 Prediction of BSI

Several other research groups have built machine learning models to predict various factors of bloodstream infection. In past research, our group has built predictive models of bloodstream with the goal of characterizing common predictors and signals of bloodstream infection, rather than to predict it. Zimmet, et al. used simple predictive models to identify distinct signatures of various types of bloodstream infection [21]. In our study, we built on these results to devise predictive models with the goal of achieving continuous monitoring as a decision aid for clinicians in the ICU.

Pai, et al. achieved very predictive models, primarily using gradient boosting and random forest search [15]. Our study is distinct from theirs in several ways: first, we included patients who had no blood cultures drawn in our dataset. Second, in their data preprocessing stage, they reduced each bloodstream infection to the means of each feature. We kept the timeseries structure of the episodes intact. Third, we applied deep learning techniques, whereas they focused on traditional machine learning approaches, such as random forests and XGBoost. Zoabi, et al. applied similar modeling techniques that were successful in predicting patient outcomes, given that the patient had a positive blood culture [22]. Our research hypothesis differed from both of these studies; in our study, we did not bias the model on the assumption that a blood culture had already been drawn or that the patient already had a positive result.



(a) Examples that match clinical expectation



(b) Examples that do not match clinical expectation

Figure 4: Fold risk predictions for 24 hour windows swept over a 48 hour period for individual patients

5.2 Multivariate Timeseries Classification

There are several approaches to multivariate timeseries classification (MVTC), the problem setting in which we operated. The first of such approaches are ensembles of univariate timeseries classifiers, such as ROCKET [8] and HIVE-COTE2 [13]. Both of these approaches show strong results on common datasets [18], but are not effective for our problem. They do not scale well in time complexity with the size of data and struggle to learn complex relationships in large datasets. Because of the size of our dataset, we were unable to compare these models to our deep learning approaches due to training time constraints.

The second approach is to apply data preprocessing techniques to reduce a multivariate timeseries classification problem to a simpler tabular data classification problem. This approach is used by Zoabi, et al [22]. They processed their data by taking the episode-wise mean of each feature. This approach allowed them to use traditional ML techniques, such as random forests and XGBoost. This approach help them achieve good results in a static setting, but was not useful

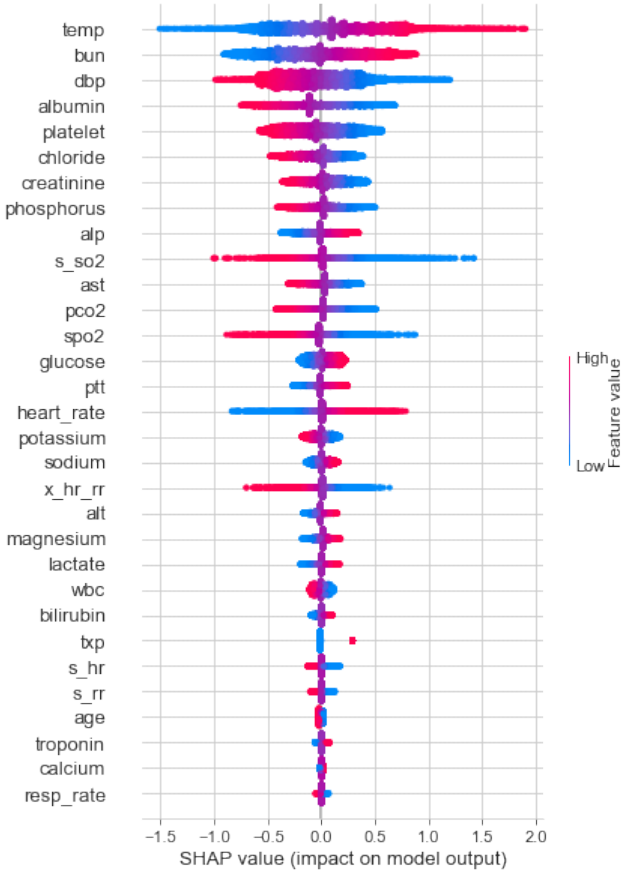


Figure 5: Shapley values for our variables evaluated from a GRU model displayed as a beeswarm plot. Descriptions of each variable are included in appendix B.

for us given our goal of achieving a machine learning model capable of continuous monitoring.

The third approach, which we focused on in our study, is to use deep learning methods to learn on the data in its raw, multivariate timeseries format. These approaches have a few key advantages: they can reason over both temporal and feature space, they scale well in time complexity with the size of the data, and they have been demonstrated to achieve very strong predictive power on other difficult timeseries classification tasks [6] [20]. In our study, we had both the amount of data and computing resources to effectively utilize deep learning approaches.

5.3 Deep Learning for Timeseries Problems

Deep Learning has recently been very successful in tackling multivariate timeseries problems in other domains. For example, Bowes, et al. recently demonstrated the predictive power of RNNs to forecast the groundwater table in flood prone coastal cities [4]. They apply their deep learning timeseries forecasting techniques to ground water, rainfall, and sea level timeseries in a multivariate timeseries dataset.

The Great Multivariate Timeseries Bakeoff examined the predictive power of a variety of approaches [18], including deep learning, on a set of 30 datasets and found that deep learning approaches compete with the very best non-deep learning approaches even on small datasets. Deep learning approaches beat traditional machine learning approaches on larger datasets. Our dataset was larger than any of the 30 used, so we expected deep learning to outperform traditional timeseries classification approaches.

In their review paper, Fawaz, et al. found that convolutional methods, such as deep residual networks and fully convolutional networks, achieve the current state of the art results on a wide variety of multivariate timeseries classification problems [12]. They found that these deep neural networks scaled better in both number of example and length of timeseries than other approaches. They also found that, in order to achieve state of the art performance with deep learning models, large amounts of data are needed. We have a sufficiently large amount of data to achieve the state of the art performance described by their survey.

6 CONCLUSION

Our work demonstrates that deep learning models show strong predictive power in the setting of bloodstream infection and paves the way for trials to evaluate its effectiveness in reducing the volume of blood cultures drawn and thereby reducing mortality due to bloodstream infection in the ICU. The continuous monitoring capability of our model could provide doctors and nurses the ability to better determine when to draw blood cultures. Our modeling approach also provides explanations to help build clinical trust and inform clinical decision making.

ACKNOWLEDGMENTS

This work was supported by a University of Virginia Global Infectious Diseases Institute Seed Grant and a University of Virginia Center for Engineering in Medicine Seed Grant Program, which were both awarded jointly to CCM and NRN.

REFERENCES

[1] David W. Bates et al. 1990. Predicting bacteremia in hospitalized patients: a prospectively validated model. *Annals of internal medicine* 113, 7 (1990), 495–500.
[2] David W. Bates, Lee Goldman, and Thomas H. Lee. 1991. Contaminant blood cultures and resource utilization: the true consequences of false-positive results. *Jama* 265, 3 (1991), 365–369.
[3] Lukas Biewald. 2020. Experiment Tracking with Weights and Biases. https://www.wandb.com/ Software available from wandb.com.
[4] Benjamin D. Bowes, Jeffrey M. Sadler, Mohamed M. Morsy, Madhur Behl, and Jonathan L. Goodall. 2019. Forecasting Groundwater Table in a Flood Prone Coastal City with Long Short-term Memory and Recurrent Neural Networks. *Water* 11, 5 (2019). https://doi.org/10.3390/w11051098
[5] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, 1724–1734. https://doi.org/10.3115/v1/D14-1179
[6] Edward Choi et al. 2016. *Doctor ai: Predicting clinical events via recurrent neural networks.* Machine learning for healthcare conference. PMLR.
[7] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. https://doi.org/10.48550/ARXIV.1412.3555
[8] Angus Dempster, François Petitjean, and Geoffrey I. Webb. 2020. ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery* 34, 5 (jul 2020), 1454–1495. https://doi.org/10.1007/s10618-020-00701-z

[9] Timothy Dozat. 2016. Incorporating Nesterov Momentum into Adam.
[10] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. 2020. InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery* 34, 6 (sep 2020), 1936–1962. https://doi.org/10.1007/s10618-020-00710-y
[11] Jessica L. Guidi et al. 2015. Clinician perception of the effectiveness of an automated early warning and response system for sepsis in an academic medical center. *Annals of the American Thoracic Society* 12, 10 (2015), 1514–1519.
[12] Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. 2019. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery* 33, 4 (July 2019), 917–963.
[13] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony Bagnall. 2021. HIVE-COTE 2.0: a new meta ensemble for time series classification. https://doi.org/10.48550/ARXIV.2104.07551
[14] Joseph Randall Moorman et al. 2011. Mortality reduction by heart rate characteristic monitoring in very low birth weight neonates: a randomized trial. *The Journal of pediatrics* 159, 6 (2011), 900–906.
[15] Kai-Chih Pai, Min-Shian Wang, Yun-Feng Chen, Chien-Hao Tseng, Po-Yu Liu, Lun-Chi Chen, Ruey-Kai Sheu, and Chieh-Liang Wu. 2021. An Artificial Intelligence Approach to Bloodstream Infections Prediction. *Journal of Clinical Medicine* 10, 13 (2021). https://doi.org/10.3390/jcm10132901
[16] Benedek Rozemberczki, Lauren Watson, Péter Bayer, Hao-Tsung Yang, Olivér Kiss, Sebastian Nilsson, and Rik Sarkar. 2022. The Shapley Value in Machine Learning. https://doi.org/10.48550/ARXIV.2202.05594
[17] Kristina E. Rudd et al. 2020. Global, regional, and national sepsis incidence and mortality, 1990–2017: analysis for the Global Burden of Disease Study. *The Lancet* 395, 10219 (2020), 200–211.
[18] Alejandro Pasos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. 2021. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery* 35, 2 (March 2021), 401–449.
[19] Mark E. Rupp et al. 2017. Reduction in blood culture contamination through the use of initial specimen diversion device. *Clinical Infectious Diseases* 65, 2 (2017), 201–205.
[20] Benjamin Shickel et al. 2017. Deep EHR: a survey of recent advances in deep learning techniques for electronic health record (EHR) analysis. *IEEE Journal of biomedical and health informatics* 22, 5 (2017), 1589–1604.
[21] Alex N. Zimmet et al. 2020. Pathophysiologic Signatures of Bloodstream Infection in Critically Ill Adults. *Critical care explorations* 2, 10 (2020).
[22] Yazeed Zoabi, Orli Kehat, Dan Lahav, Ahuva Weiss-Meilik, Amos Adler, and Noam Shomron. 2021. Predicting bloodstream infection outcome using machine learning. *Scientific Reports* 11, 1 (Oct. 2021), 20101.

A MODEL ARCHITECTURES

Table 3: Architecture of CNN with highest validation AUC

	Type	Parameters	Output Shape
0	InputLayer	0	,48,33,1
1	Conv2D	320	None, 48, 33, 32
2	Conv2D	9248	None, 48, 33, 32
3	MaxPooling2D	0	None, 24, 17, 32
4	Conv2D	18496	None, 24, 17, 64
5	Conv2D	36928	None, 24, 17, 64
6	MaxPooling2D	0	None, 12, 9, 64
7	Conv2D	73856	None, 12, 9, 128
8	Conv2D	147584	None, 12, 9, 128
9	MaxPooling2D	0	None, 6, 5, 128
10	Flatten	0	None, 3840
11	Dense	983296	None, 256
12	Dense	257	None, 1

Table 4: Architecture of GRU with highest validation AUC

	Type	Parameters	Output Shape
0	InputLayer	0	,48,33
1	Masking	0	None, 48, 33
2	GRU	6432	None, 48, 32
3	BatchNormalization	128	None, 48, 32
4	Dropout	0	None, 48, 32
5	GRU	18816	None, 48, 64
6	BatchNormalization	256	None, 48, 64
7	Dropout	0	None, 48, 64
8	Flatten	0	None, 3072
9	Dense	3073	None, 1

Table 5: Architecture of Feed Forward Baseline

	Type	Parameters	Output Shape
0	InputLayer	0	,48,33
1	Masking	0	None, 48, 33
2	BatchNormalization	132	None, 48, 33
3	Flatten	0	None, 1584
4	Dense	1585	None, 1

B FEATURES

Feature abbreviations and human readable names:

temp: Core Body Temperature
bun: Blood Urea Nitrogen
dbp: Diastolic Blood Pressure
albumin: Albumin
platelet: Platelet Count
chloride: Chloride
creatinine: Creatinine
phosphorus: Phosphorus
alp: Alkaline Phosphate
s_so2: Supersaturated Oxygen
ast: Aspartame Aminotransferase
pco2: Partial Pressure of Carbon Dioxide
spo2: Oxygen Saturation
glucose: Blood Glucose
ptt: Partial Thromboplastin Time
heart_rate: Heart Rate
potassium: Potassium
sodium: Sodium
x_hr_rr: Cross Correlation of Heart Rate and Respiratory Rate
alt: Alanine Aminotransferase
magnesium: Magnesium
lactate: Lactate
wbc: White Blood Cell Count
bilirubin: Total Bilirubin
txp: Patient is a Transplant Recipient
s_hr: Standard deviation of Heart Rate
s_rr: Standard deviation of Respiratory Rate
age: Age
troponin: Troponin
calcium: Calcium
resp_rate: Respiratory Rate