The impact of time-series preprocessing and deep-learning architectures for early sepsis detection

Patrick Burgård Bjerre
Software Engineering
University of Southern Denmark

***Abstract*—Sepsis is a life-threatening condition where early detection is critical for patient survival. Yet, it remains challenging due to the high sparsity and extreme class imbalance of clinical time-series data. This study evaluates the impact of data pre-processing strategies and Deep Learning architectures for early sepsis prediction using the PhysioNet 2019 Challenge dataset. We investigate three distinct architectures across four pre-processing methods designed to handle the approximate 80% missingness in hourly vital signs.**

Experimental results demonstrate that a two-layer LSTM architecture significantly outperforms baseline models, particularly when utilizing a 'Negative-1' strategy that explicitly encodes data missingness as a feature. While the initial grid search yielded a best Utility Score of 0.2015, a subsequent threshold analysis calibrated for clinical sensitivity improved the Utility Score to 0.3396, while maintaining a robust AUC of 0.99. These findings suggest that while deep learning models can effectively extract temporal features from sparse medical data, their clinical deployment requires rigorous calibration of decision thresholds to balance the asymmetric costs of false negatives and false positives.

***Index Terms*—Sepsis, Time Series Analysis, Long Short-Term Memory, Recurrent Neural Network**

I. Introduction

Sepsis affects an estimated 46 million individuals each year, resulting in approximately 11 million fatalities. Those most at risk are the elderly, pregnant women, and hospitalized patients. Advancing early detection methodologies in intensive care units (ICUs) can reduce mortality in a high-risk patient group. Sepsis is a disease with a clear definition and a list of symptoms, such as a heart rate > 90 bpm, a lower white cell count, and a higher respiratory rate. These definitions are consistent with the sepsis-3 definition [1], but for an ICU patient, it can also be something else causing these numbers. The cost of treating every patient for sepsis is high for the health care system. It can lead to complications depending on the patient and their other needs. Therefore, getting the earliest possible hint of sepsis would make it easier to treat and, likely, lower the mortality rate. For the elderly who have suffered from sepsis, their 2-year mortality rate becomes close to 50%, [2] showing the heavy burden it can have on a person with an already burdened immune system. The challenge with early sepsis detection is that, even with the definition, it is just a guideline for determining whether a patient already has sepsis, and symptoms can overlap with other possible health problems, making it unclear before it can be confirmed. Here, deep learning models can help in the research of the infection. By looking at the entire health picture of the patient, values outside normal definition can be used and tailored to specific patients better, given that a proportional increase in heart rate and respiratory rate, as well as white cell count, could indicate sepsis even if the patient is not above the vital thresholds, as per the definition.

In 2019, a challenge from the website PhysioNet was proposed to detect this sepsis [3]. The aim was to see whether any ideas from the community could help solve this critical issue. They supplied the data and the case, and this will serve as the use case for this project, which will develop an analysis using deep learning techniques for time series. This will take into account three different model architectures and see which is best at predicting the earliest. The model architectures are Convolutional neural networks (CNNs), Recurrent neural networks (RNNs), and Long Short-Term Memory (LSTM) models. The models will be evaluated using a grid search to determine the best version, and the best model will then be tested on both the AUC and the physnet2019 Utility score. Different pre-processing methods for time series analysis will also be tested to assess their impact on model performance. The data supplied from physionet contains sepsis data from 3 hospitals, labeled A, B, and C. Only A and B were released as part of the challenge, and C was used for the challenge's blind scoring. The aim of the project is to identify ways deep learning can help improve the space, due to the benefits deep learning models can have on a large time series dataset, where it can learn the features and help determine, on a patient-by-patient basis, if there are indications that other classification methods would miss.

II. Related Work

Due to the fact that the dataset and case were presented as a challenge, the papers written for it were submitted to a journal. Given that the challenge has now completed, the papers are available. The best-ranked model, with the best utility score of 0.43 (1 is perfect), used a forward fill method to handle missing data, and a signature-based model [4]. It also introduced some new features based on the others to better interpolate them. This left the team with the best-performing model for the entire challenge. The second-ranked model used a decision tree approach with gradient boosting, and for pre-processing, used a form of a Gaussian distribution with zero values and forward fill from the first valid value [5]. This team scored almost 0.4 utility score in their evaluation. Many of the top-scoring models used decision trees and gradient boosting [6], and the importance of their pre-processing methods cannot be overstated; the hand-picked, hand-crafted approach to pre-processing in the top models is very likely to have contributed to their high scores. One of the top-scoring teams tried both a tree-based result and a deep learning result using LSTM. Their LSTM-based model scored a 0.256 while the tree-based XDGBoost scored closer to 0.4 [7]. The team claims the imbalance of the dataset is the reason the LSTM model scores poorly. Knowing the state of the scores and how it was approached by other people, and using different model architectures, this project will try to explore deep learning architectures to see if deep learning can be used to extend this area of research, using different approaches to see if this can outperform the existing tree-based or improve on the deep learning based ones. Deep learning variants can learn the features, so the amount of feature engineering that the other top-scoring groups have performed will be limited. This could help if feature engineering becomes too expensive an operation, if the dataset is extended, or to limit the required interpolation of the data, and just use a model that will learn the necessary features.

III. Method

Three distinct model architectures will be evaluated in this project.

The first architecture is a convolutional neural network (CNN) consisting of two one-dimensional convolutional layers, each followed by a ReLU activation function and a dropout layer, with the dropout rate treated as a hyperparameter. These layers will be followed by a fully connected (dense) layer that computes a final score over all extracted features. This score will then be passed through an output activation function, which will also be selected as a hyperparameter. Additional hyperparameters for the CNN model will include filter size, kernel size, and stride.

The recurrent neural network (RNN) model will consist of a recurrent layer that processes hourly data and, after iterating through the sequence, encodes the entire temporal history in its hidden state. A dropout layer will be used, with the dropout rate treated as a hyperparameter, as will the choice of the final activation function.

The long short-term memory (LSTM) model will comprise two recurrent layers. This structure is intended to enable hierarchical temporal feature learning (i.e., "features of features"), such as the detection of rapid changes in signals, including heart rate or blood pressure.

To get a better overview of the model architecture, see figure(3)

A. Scoring

The challenge is early detection, so a utility function has been developed by PhysioNet. The utility function will penalize the model for a late prediction where the sepsis is already being treated. The formula for the utility function was provided by the challenge organizers and can be seen equation 1. A scoring function was created to operate within these parameters and used, along with an AUC score as a tiebreaker, to assess the viability of the models.

$$U(t) = \begin{cases} \frac{t-t_{onset}+12}{6} & \text{if } -12 \leq t_{diff} \leq -6 \\ 1.0 & \text{if } -6 < t_{diff} \leq 3 \\ -0.05 & \text{if } \hat{y} = 1 \text{ outside optimal window} \\ 0 & \text{if } \hat{y} = 0 \end{cases} \tag{1}$$

IV. Experiments

To evaluate different configurations of these models and identify the best ones, a multi-stage approach will be used.

A. Data preprocessing

The dataset provided by the PhysioNet challenge is sourced from two separate hospitals, totaling around 40,000 patients [3]. As seen in Figure 2, the data has extreme class imbalance, with sepsis cases representing only around 7% of ,the total population. Furthermore, the data density is approximately 20%, implying that 80% of the hourly vital signs are missing (e.g., a patient might have heart rate recorded every hour, but lactate levels only once a day).

To address this imbalance, we evaluate four distinct preprocessing strategies to determine which has the highest impact on model performance:

1) **Negative-1 (neg1):** All missing entries are replaced with a value of -1. Since the measurements are all positive values, the negative values allow for a clear distinction between an existing value and missing data.
2) **Mean:** Missing values are filled with the patients average. If a patient has no data for a feature, the global mean is used.
3) **Median:** Similar to mean, but using the median value.
4) **Linear Interpolation (Linear):** Missing values are approximated by drawing a straight line between the last known value and the next known value. If no data is collected, the value -1 is used.

This pre-processing only accounts for missing data, but the imbalance remains. This imbalance could be helped using oversampling, but due to the risk of overfitting to specific patients' statistics, it was determined to let the imbalance become a topic for future analysis.

The data is split up in an 80/10/10 split. There will be a prediction every hour. To limit the extremes of a few patients with an extended length of stay. This will be capped to 256 hours to limit computational complexity and reduce training time by removing the outliers while keeping most of the data.

B. Grid Search

The next part is the grid search. The grid search has some parameters in common, like loss functions and dropout. *The batch size was reduced to 8-16 for the LSTM later in the process because it was using too much VRAM on the GPU*

Generative AI was used to assist in the creation of the experimental code.

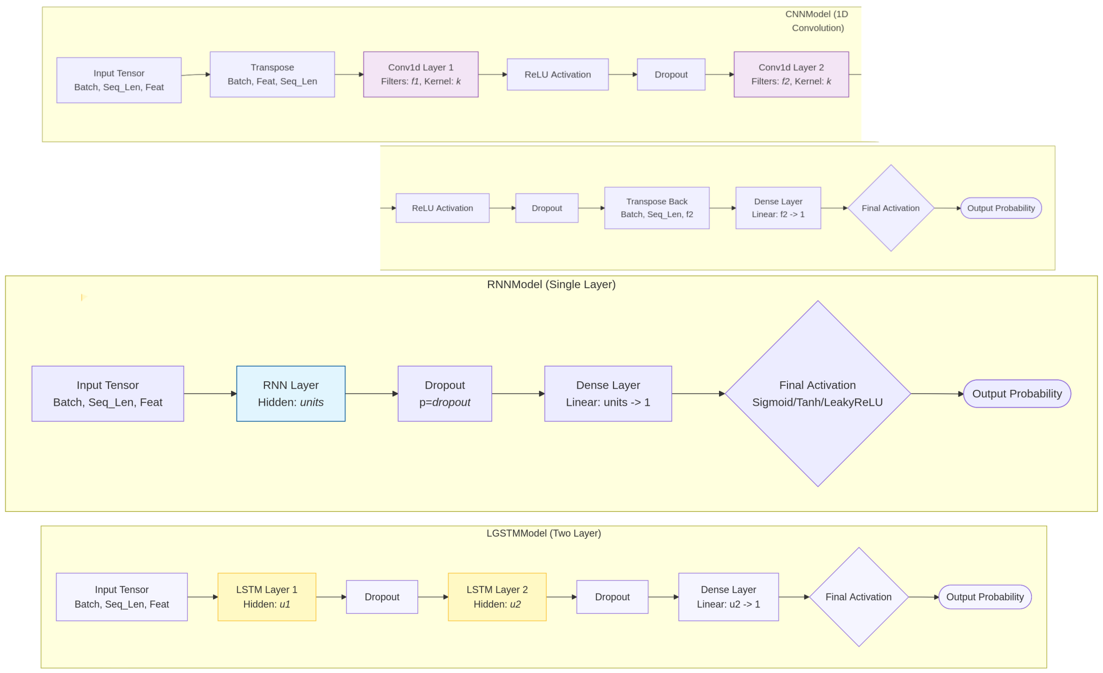
CNNModel (1D Convolution)
Input Tensor Batch, Seq_Len, Feat
Transpose Batch, Feat, Seq_Len
Conv1d Layer 1 Filters: f1, Kernel: k
ReLU Activation
Dropout
Conv1d Layer 2 Filters: f2, Kernel: k
ReLU Activation
Dropout
Transpose Back Batch, Seq_Len, f2
Dense Layer Linear: f2 -> 1
Final Activation
Output Probability
RNNModel (Single Layer)
Input Tensor Batch, Seq_Len, Feat
RNN Layer Hidden: units
Dropout p=dropout
Dense Layer Linear: units -> 1
Final Activation Sigmoid/Tanh/LeakyReLU
Output Probability
LGSTMModel (Two Layer)
Input Tensor Batch, Seq_Len, Feat
LSTM Layer 1 Hidden: u1
Dropout
LSTM Layer 2 Hidden: u2
Dropout
Dense Layer Linear: u2 -> 1
Final Activation
Output Probability

Fig. 1. Model Architectures

Hospital system	A	B
Number of patients	20,336	20,000
Number of septic patients	1,790	1,142
Sepsis prevalence	8.8%	5.7%
Number of rows	739,663	684,508
Number of entries	5,536,849	4,950,064
Density of entries	20.6%	19.1%

Fig. 2. The Datasets distribution, density and imbalance [3]

used for training. The Full list of grid search parameters can be seen in the table I

TABLE I
HYPERPARAMETER GRID SEARCH SPACE

Common Hyperparameters	**Search Values**
Optimizer	$\{\text{Adam}, \text{AdamW}\}$
Learning Rate	$\{0.001, 0.0005\}$
Dropout Rate	$\{0.2, 0.4\}$
Loss Function	$\{\text{Binary Cross-Entropy}, \text{MSE}, \text{Hinge}\}$
Final Activation	$\{\text{Sigmoid}, \text{LeakyReLU}, \text{ELU}, \text{Tanh}\}$
RNN Specific	
Batch Size	$\{32, 64\}$
Hidden Units	$\{64, 128\}$
CNN Specific	
Batch Size	$\{32, 64\}$
Layer 1 Filters (f_1)	$\{32, 64\}$
Layer 2 Filters (f_2)	$\{64, 128\}$
Kernel Size	$\{3, 5\}$
Stride	$\{1, 2\}$
LSTM Specific	
Batch Size	$\{8, 16\}$
Layer 1 Units (u_1)	$\{64, 128\}$
Layer 2 Units (u_2)	$\{32, 64\}$

C. *Training procedure*

Initially, a grid search will be performed for each pre-processing method using 15 training epochs. For each architecture, the models corresponding to the two highest Utility scores and the two highest AUC scores will then be selected. Unless there is configuration overlap, this selection trains 12 top-performing models for each preprocessing strategy. These models will then be trained for 25, 50, and 100 epochs in order to investigate potential correlations between the number of epochs and model performance. Finally, the best-performing model from this stage will be further trained with epoch counts increasing in steps of 25 until a maximum of 300 epochs is reached. This will gain an insight into what model architectures and what pre-processing methods will be useful to further analyze.

V. RESULTS AND DISCUSSION

All training was performed on a single desktop using an NVIDIA RTX 5060 TI. This was implemented using the PyTorch framework and the CUDA framework for GPU parallelization.

A. *Preprocessing*

The data pre-processing resulted in the negative one version, which showed the greatest outcome; this corresponds to one of the challenge entries focusing on the relevance of the missingness of the data [8]. The worst performing method was the linear interpolation, which suggests that putting in data that may not be relevant can confuse the models. The mean performed better than the median and linear, possibly due to a global average becoming a missingness feature. Interestingly, the AUC score was mean, and the median was better than neg1, indicating that early detection and detection are not always correlated. Preprocessing These findings suggest that the absence of observations is nearly as informative as the observed data themselves. Interpolating missing values may artificially inflate performance metrics such as the AUC, while also degrading early detection capability, particularly when such pre-processing is applied before model training.

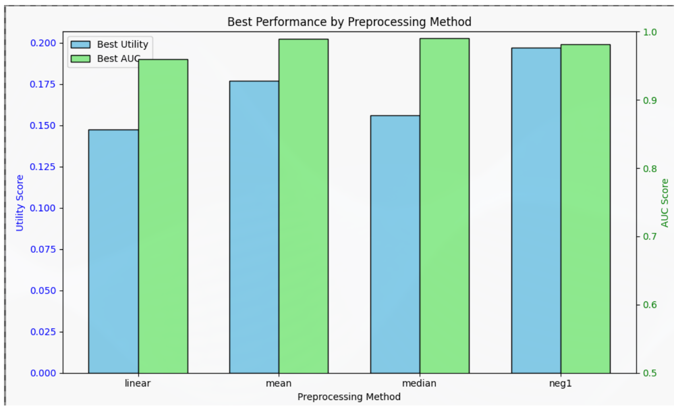

Fig. 3. Bar chart showing utility and AUC scores of pre-processing methods after grid search. AUC scores shown as percentages.

B. Architecture

After the grid search, the top-performing models were all trained for 100 Epochs. After the full training, a comparison of the best score of each architecture was conducted, as shown in Table II. It seemed like the LSTM model was the only one capable of fully leveraging the missingness of negative values. This is likely due to the memory of the model, which can take it into account. The CNN architecture scored a high AUC but had a poor utility score, this indicates that even it can see a clear difference between sepsis patients and non sepsis patients, the early prediction fails. The RNN architecture scores poorly in both; this suggests that the ability to store more long-term memory, like the LSTM model, is a significant factor for early detection and detection altogether. This corresponds with the idea that even though sepsis has clear guidelines in terms of, for example, heart rate, on the individual patient level, an increase or decrease of a level can also have a significant role. It was clear that the LSTM model architecture was the best, as it ranked highest across all metrics; the next analysis will focus only on the LSTM. The best-performing model configuration is shown in Table III. With a best score of 0.197, the LSTM Utility-focused model performed the best. Interestingly, the runner-ups were chosen for their high AUC score before being trained for more epochs. There is a further increase in epochs that could influence the score in theory.

C. Epoch analysis and fine tuning

The three best-performing configurations shown in Table III have been trained with 25-epoch increments to assess the

TABLE II
CROSS-ARCHITECTURE PERFORMANCE COMPARISON

Arch.	Type	Util ↑	AUC	Drop	LR	Loss
LSTM	neg1	**0.1970**	**0.9815**	0.4	1e-3	mse
CNN	median	0.1044	0.9771	0.2	1e-3	bce
RNN	median	0.0769	0.7744	0.2	1e-3	mse

TABLE III
TOP PERFORMANCE CONFIGURATION

Model ID	Type	Util ↑	AUC	Drop	LR	Units
LSTM_Util_199	neg1	**0.1970**	0.9815	0.4	1e-3	128/64
LSTM_AUC_134	mean	0.1770	0.9897	0.4	1e-3	128/32
LSTM_AUC_14	mean	0.1666	0.9891	0.2	5e-4	128/32

impact of different epochs on model performance; the highest epoch was 300. The result of the epoch training can be seen in Figure 5. The models lacked a clear epoch progression, suggesting that the randomness made them unstable at this number of epochs. It is possible that a higher epoch would stabilize them. It could also be a possibility that the imbalance in the dataset makes it hard for the model to gain a clear features of features and the epochs iteration finds different features of features or muddies the existing ones. Another curious feature of the epoch analysis is that the AUC score, figure 6, seems to be stable in the models that scored high on the initial grid search, but also seems to be confusing the AUC score of the one that had the best Utility. A new best model was also found. The runner-up scored extremely well on the 300 epochs, which made the new best score **0.2015**. Surprisingly, that model used the mean pre-processing tactic. This could mean that the features of features work better with an average than with the missingness values, which scored better on small epochs.
Even though the model scores around the state of the art for the LSTM model, improvements can still be made. One approach that was tried late in the process is to try different thresholds for what constitutes a sepsis detection. The idea came from looking at the confusion matrix 7 and seeing it had an almost equal likelihood of getting a false positive and a false negative. In this case, with this deadly disease, getting a model to have higher false positives than false negatives seemed paramount.

D. threshold analysis

The threshold analysis was run with an epoch of 75, because from the analysis seen in figure 1, it was the high-scoring and stable point of the training, giving each model a chance of showing a good number while not being too computationally expensive. Running a threshold analysis clearly shows, as seen on figure 8, that the lower the threshold, the better the utility score. This also raises the amount of false positives the model makes, but the risk of a false negative is much greater than a false positive. The same model configuration still scored the best, but this time with a score of 0.2685, now outscoring the existing LSTM model that was entered in the competition.

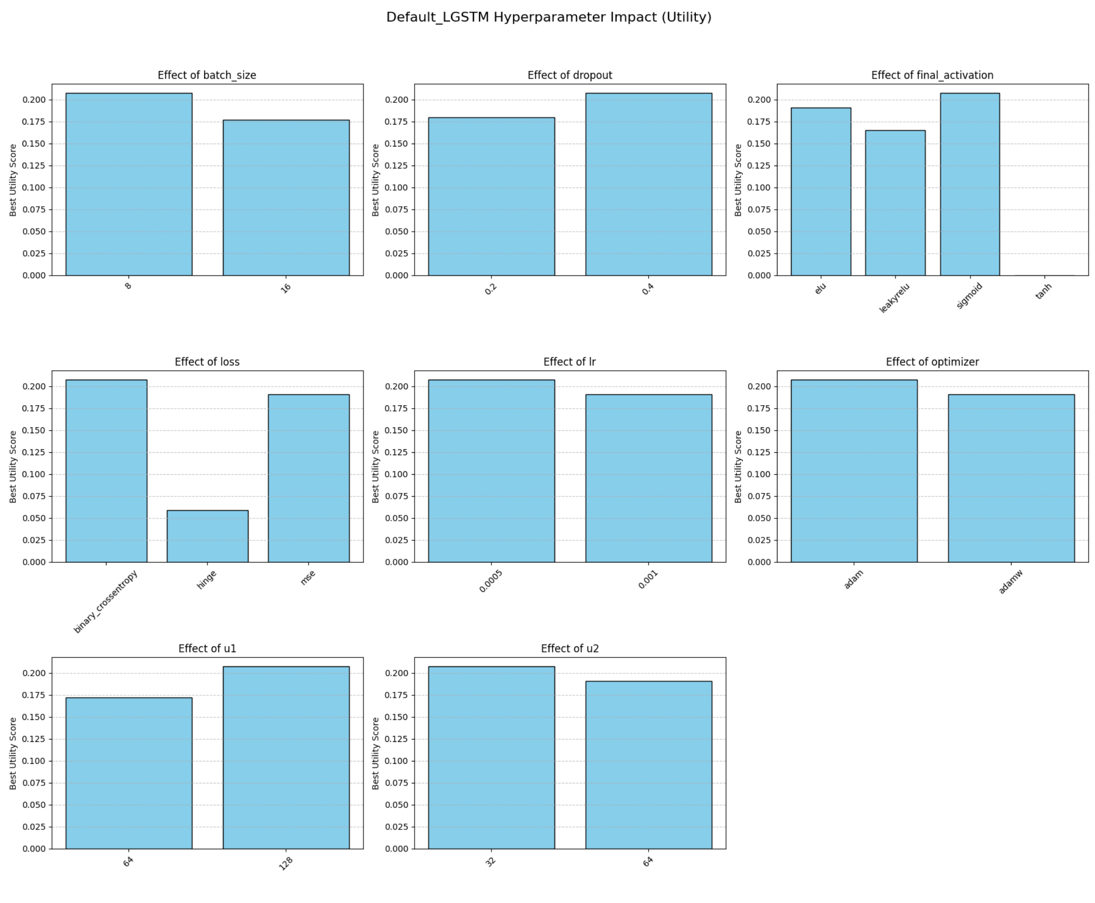

Fig. 4. A full view of the hyperparameter impact on LSTM models

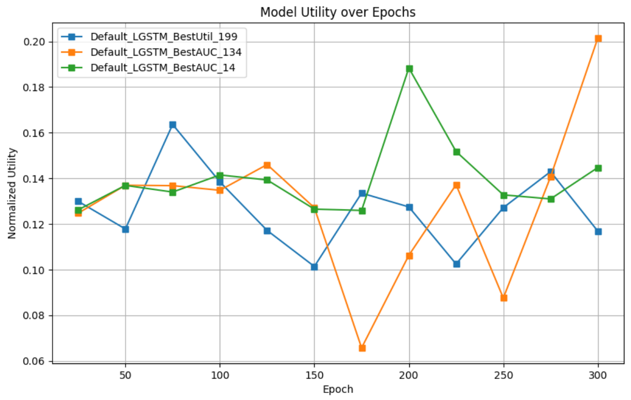

Fig. 5. Model Utility score over Epochs

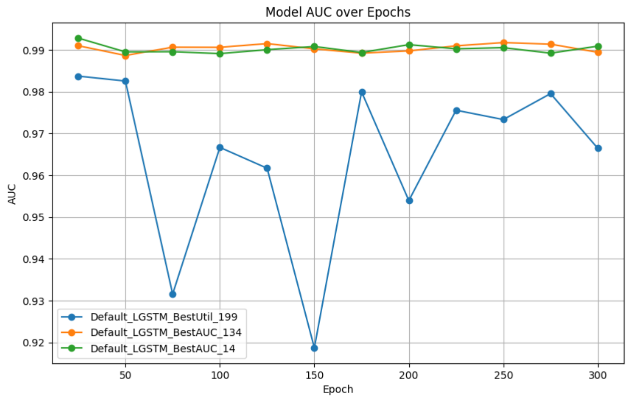

Fig. 6. Model AUC score over Epochs

This threshold analysis shows that just a small indication of sepsis should be enough to be flagged as a possible sepsis. If it takes even further to a threshold of 0.005, which is the smallest before a model collapse, and also gave a Utility score of 0.3396, see 9. The almost 8000 false positives is an issue, but the utility function highlighted by the challenge does not penalize false positives, as hard as it rewards early predictions. It is assumed that this is a feature of the challenge and the cost of being wrong but cautious is lower than the probability of mortality resulting from the infection.

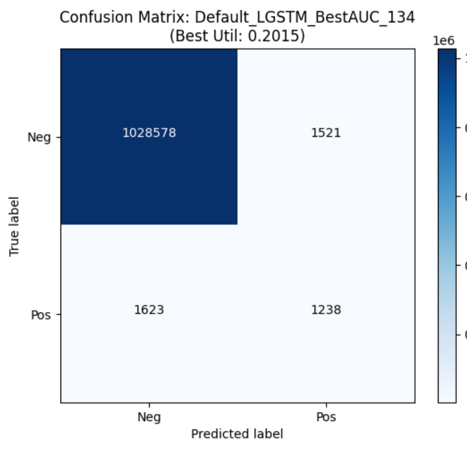

Fig. 7. Confusion Matrix From the best performing model

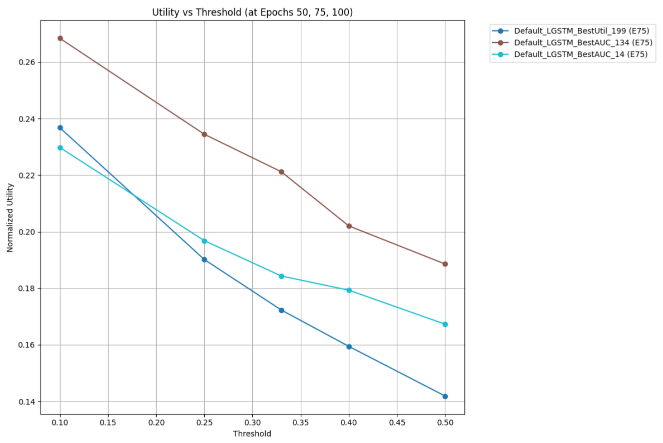

Fig. 8. A threshold analysis of the utility score

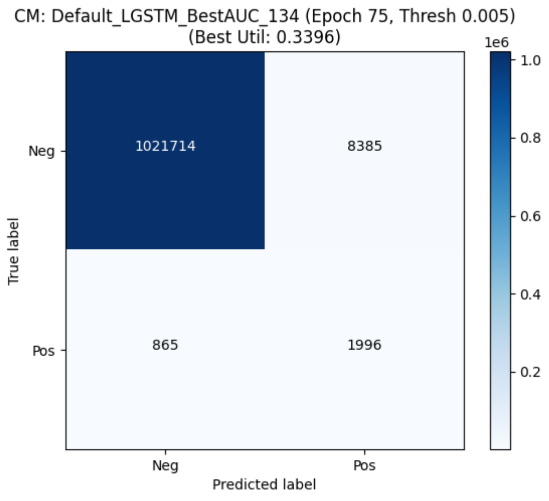

Fig. 9. Extremely low threshold, but has a high false negative rate

E. Possible improvements and future work

Even though the final model had a utility score of 0.3396 and an AUC of 0.990, there were several things that could be improved to get an even better model. The imbalance is still an improvement that a lot could be gained from, and oversampling seems like a straightforward case.
Using a lower threshold for the grid search could have been beneficial. There could also be a larger grid search to find better configurations, since even though it has a respectable search space, the training was done on a single consumer-grade desktop. For real clinical use cases, a larger grid space, training time, and computing in general would be beneficial.
For the analysis, it could have been useful to have used an F1 or a recall score. This was not implemented due to the early detection being the utility score. But a view into the false negatives when lowering the threshold and getting the best cost-benefit for clinical use cases would have been preferred.
It could also be interesting to look into mixing the two best-performing pre-processing methods with a missingness attribute from neg1 instead of the global average. The global average could be misleading in some cases, and showing missingness might be a feature for better data understanding.

VI. Conclusion

A model has been made that does not outcompete the existing tree-based models, but does show that deep learning models can be part of the state of the art in early sepsis detection. It has shown that it outperforms other attempts at making an LSTM model, while still balancing a high AUC score. It was shown that the best performing pre-processing method was neg1, which highlights missingness in the data, and mean, which uses a mix of per-patient averages and global averages, depending on the number of epochs. It also proves that for early detection, an AUC score is not always indicative of a perfect model. It also showed the superiority of the LSTM architecture over CNN and RNNs.

References

[1] M. Singer, C. S. Deutschman, C. W. Seymour, M. Shankar-Hari, D. Annane, M. Bauer, R. Bellomo, G. R. Bernard, J.-D. Chiche, C. M. Coopersmith, R. S. Hotchkiss, M. M. Levy, J. C. Marshall, G. S. Martin, S. M. Opal, G. D. Rubenfeld, T. van der Poll, J.-L. Vincent, and D. C. Angus, “The Third International Consensus Definitions for Sepsis and Septic Shock (Sepsis-3),” *JAMA*, vol. 315, no. 8, pp. 801–810, Feb. 2016. [Online]. Available: https://doi.org/10.1001/jama.2016.0287

[2] F. Nielsen, L. Chafranska, R. Sørensen, T. Schmidt, and O. Abdullah, “Two-Year Mortality and Prognostic Factors in Sepsis: A Prospective Cohort Study of 714 Danish Emergency Department Patients,” *Clinical epidemiology*, vol. 17, no. Issue 1, pp. 581–592, 2025, place: New Zealand Publisher: Dove Medical Press Limited.

[3] M. A. Reyna, C. S. Josef, R. Jeter, S. P. Shashikumar, M. B. Westover, S. Nemati, G. D. Clifford, and A. Sharma, “Early Prediction of Sepsis From Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019,” *Critical Care Medicine*, vol. 48, no. 2, pp. 210–217, Feb. 2020. [Online]. Available: https://journals.lww.com/10.1097/CCM.0000000000004145

[4] J. Morrill, A. Kormilitzin, A. Nevado-Holgado, S. Swaminathan, S. Howison, and T. Lyons, “The Signature-Based Model for Early Detection of Sepsis from Electronic Health Records in the Intensive Care Unit,” Dec. 2019. [Online]. Available: http://www.cinc.org/archives/2019/pdf/CinC2019-014.pdf

[5] J. Anda Du, N. Sadr, and P. De Chazal, “Automated Prediction of Sepsis Onset Using Gradient Boosted Decision Trees,” Dec. 2019. [Online]. Available: http://www.cinc.org/archives/2019/pdf/CinC2019-423.pdf

[6] “Early Prediction of Sepsis from Clinical Data: The PhysioNet/Computing in Cardiology Challenge 2019.” [Online]. Available: https://moody-challenge.physionet.org/2019/results/

[7] Y. Wang, B. Xiao, X. Bi, W. Li, J. Zhang, and X. Ma, “Prediction of Sepsis from Clinical Data Using Long Short-Term Memory and eXtreme Gradient Boosting,” Dec. 2019. [Online]. Available: http://www.cinc.org/archives/2019/pdf/CinC2019-192.pdf

[8] J. Singh, K. Oshiro, R. Krishnan, M. Sato, T. Ohkuma, and N. Kato, “Utilizing Informative Missingness for Early Prediction of Sepsis,” Dec. 2019. [Online]. Available: http://www.cinc.org/archives/2019/pdf/CinC2019-280.pdf